Soumission

Journal de la Société Française de Statistique

Représentation d'un grand réseau à partir d'une classification hiérarchique de ses sommets

Title: Large graph visualization from a hierarchical node clustering

Fabrice Rossi[1] et Nathalie Villa-Vialaneix[1]

Résumé : Les graphes (ou réseaux) sont devenus des outils courants de modélisation des données relationnelles dans de nombreuses applications (réseaux sociaux, biologiques, informatiques...). Or, lorsque le nombre de sommets dépasse quelques centaines, la visualisation du graphe dans son ensemble, qui est un outil important de compréhension du réseau, est un problème complexe : les approches traditionnelles, basées sur des algorithmes de forces, s'avèrent coûteuses en temps de calcul et ne mettent pas bien en valeur la structure du réseau en parties denses (souvent appelées « communautés »). Dans cet article, nous proposons une méthode de visualisation basée sur une classification hiérarchique des sommets : cette approche permet d'obtenir des représentations de graphes de plusieurs milliers de sommets en quelques secondes, en produisant des représentations avec des niveaux de simplification plus ou moins grossiers. L'utilisateur a donc accès à des visualisations lui permettant de comprendre la structuration macroscopique du réseau puis, par zooms successifs à des détails de plus en plus fins dans chacune des communautés. La finesse maximale est contrôlée par simulation. La qualité des partitions considérées est évaluée par la mesure classique de modularité et comparée à la qualité obtenue par la méthode proposée sur des graphes aléatoires dont la distribution des degrés est identique à celle du graphe étudié : on obtient ainsi une distribution de la modularité dans le cas sans structure, ce qui permet de ne montrer que les structures significatives. Cette approche est illustrée sur plusieurs jeux de données publics et comparée à d'autres méthodes de visualisation destinées à mettre en valeur les communautés du réseau. Elle est également testée sur un grand réseau issu d'un corpus d'archives du Moyen-Âge.

Abstract: Graphs (or networks) are widely used to model relational data in various application fields (e.g., social network, biological network, Internet network...). Visualization is an important tool to understand the main features of the network but, when the number of nodes in the graph is greater than a few hundreds, standard visualization methods, such as force directed algorithms, are computationally expensive and almost unworkable. Moreover, force directed algorithms do not help the understanding of the structure of the network into dense communities of nodes, which is often a natural way to better understand a network. In this paper, a new visualization method is proposed, based on a hierarchical clustering of the nodes of the graph. This approach can handle the visualization of graphs having several thousands nodes in a few seconds. Several simplified representations of the graph are accessible, giving the user the opportunity to understand the macroscopic organization of the network and then, to focus with more details on some particular parts of the graph. This refining process is controlled by means of Monte Carlo simulation. Partitions under consideration are evaluated via the classical modularity quality measure. A distribution of the quality measure in the case of graphs without structure is obtained by applying the proposed method to random graphs with the same degree distribution as the graph under study. Then only significant partitions (with respect to this random level) are shown during the refining process. This approach is illustrated on several public datasets and compared with other visualization methods meant to emphasize the graph communities. It is also tested on a large network built from a corpus of medieval land charters.

Mots-clés : réseau, graphe, visualisation, classification, modularité

Keywords: network, graph, visualization, clustering, modularity

Classification AMS 2000 : 62-07, 62P25, 62P99

[1] Laboratoire SAMM, Université Paris 1 (Panthéon-Sorbonne)
E-mail : fabrice.rossi@univ-paris1.fr, nathalie.villa@univ-paris1.fr

Soumis au Journal de la Société Française de Statistique
File: rossi_villavialaneix_jsfds.tex, compiled with jsfds, version : 2009/12/09
date: 13 décembre 2011

1. Introduction

Les graphes (ou réseaux) sont des modélisations naturelles d'un grand nombre de données rencontrées dans des problèmes réels où les entités étudiées ne sont pas seulement décrites par des attributs numériques ou qualitatifs mais aussi par les relations qu'elles entretiennent les unes avec les autres. On retrouve, par exemple, ce type de données dans le domaine biomédical (réseaux de régulation génique, voies métaboliques, réseaux de neurones...), en informatique (réseaux d'ordinateurs, réseaux *peer to peer*...) ou, bien sûr, dans le vaste champ des réseaux sociaux. Dans nombre de ces applications, les graphes étudiés contiennent plusieurs centaines voire plusieurs milliers de sommets, ce qui rend leur compréhension et leur analyse difficiles sans l'aide d'outils de fouille de données adaptés. Aussi, les méthodes dédiées à la classification des noeuds d'un graphe (aussi appelée recherche de communautés), d'une part, et la visualisation de graphes, d'autre part, ont connu un fort développement ces dernières années.

Cependant, l'objectif de la visualisation de graphe [11, 20], qui vise à en fournir une représentation harmonieuse et agréable à l'œil, et celui de l'aide à l'interprétation peuvent être assez différents l'un de l'autre. En effet, pour comprendre la structure du graphe, l'analyste recherchera volontiers des parties denses, dans lesquelles les nœuds sont fortement connectés et qui ont peu (ou moins) de relations entre elles : cela lui permet de comprendre la structure macroscopique du réseau, avant, éventuellement de se concentrer sur des détails plus fins dans une ou plusieurs parties de celui-ci, dans une démarche proche de celle mise en œuvre dans la recherche automatique de communautés [15]. Or, la plupart des algorithmes de représentation de graphes, basés sur des modèles de forces, se concentrent sur un rendu qui favorise des arêtes courtes et de même taille ; comme le remarque Noack dans [27], cette approche a pour effet de regrouper les nœuds de forts degrés au centre de la figure, ce qui empêche l'utilisateur de les associer visuellement à des groupes différents et donc, gêne la compréhension de la structure du réseau. Une approche alternative consiste à proposer une représentation qui respecte la structure en communautés du réseau, cette structure apparaissant de manière assez naturelle dans de nombreux exemples de graphes réels.

Pour ce faire, deux grandes catégories d'approches ont été explorées : il s'agit soit de visualiser le graphe complet d'une manière qui mette en évidence sa structure de classes, soit de visualiser un graphe simplifié, en général le graphe des classes. La plupart des méthodes de la première catégorie s'appuient sur une classification des sommets du graphe obtenue au préalable et contraignent les sommets d'une même classe à être représentés à proximité les uns des autres tout en étant éloignés des sommets de classes distinctes. Ce problème est étudié depuis le milieu des années 90 sous le nom anglais de *clustered graph visualization* [5, 13, 14, 38]. Les solutions proposées s'appuient en général sur des algorithmes de forces modifiés, obtenus en intégrant les contraintes de la classification dans des modèles de forces classiques comme ceux de [12, 17].

Au sein de cette première catégorie, une approche très différente a été proposée dans [27] : elle ne passe pas par une étape de classification des sommets mais exploite directement l'optimisation d'un modèle d'énergie conçu pour représenter les sommets situés dans une zone dense du graphe à proximité les uns des autres tout en les éloignant des autres sommets.

La deuxième catégorie de méthodes propose une visualisation simplifiée du graphe, dans laquelle seules les classes sont représentées et non l'intégralité des sommets : chaque classe est symbolisée par un pictogramme donné dont la surface est proportionnelle au nombre de sommets

Soumis au Journal de la Société Française de Statistique
File: rossi_villavialaneix_jsfds.tex, compiled with jsfds, version : 2009/12/09
date: 13 décembre 2011

de la classe. Quand la classification est construite indépendamment de sa visualisation, cette technique est considérée comme classique [20]. Une approche plus originale consiste à déterminer simultanément la classification et sa visualisation, comme proposé dans [4, 35] : la classification est organisée graphiquement sur une grille en deux dimensions (type carte auto-organisatrice, par exemple).

La frontière entre ces deux catégories est floue, comme le montrent [1, 2, 37] : les auteurs utilisent une classification hiérarchique des sommets combinée à un algorithme de forces pour proposer une représentation multi-niveaux qui permet à l'utilisateur, par zooms successifs, d'accéder à des détails plus ou moins fins dans les classes. Dans cette approche, les classes sont représentées soit par des symboles soit par une représentation du sous-graphe correspondant aux sommets de la classe. La représentation est simplifiée car, notamment, seuls les liens entre classes apparaissent et non les liens individuels entre sommets de deux classes distinctes, mais l'intégralité des sommets est potentiellement visible pour l'utilisateur.

Notre proposition se rapproche des travaux [1, 2, 37] dans le sens où nous proposons de donner une représentation simplifiée et hiérarchique du graphe. Un algorithme de classification hiérarchique des sommets, utilisant une méthode éprouvée de recherche de communautés [29], est utilisé dans une première phase. Cet algorithme est très rapide et permet de traiter des graphes de plusieurs milliers de sommets en quelques secondes sur un ordinateur de bureau ordinaire ; il fournit une classification à plusieurs niveaux des sommets, chaque classe étant raffinée jusqu'à ce que sa partition ne soit plus significativement pertinente du point de vue de l'enrichissement du critère de qualité. La pertinence statistique des valeurs de la modularité est mesurée par simulation à partir du modèle de graphe aléatoire associé au critère de modularité. Le graphe induit par les communautés est alors représenté (de manière statique) avec un niveau de détails plus ou moins fin (selon le niveau de la classification hiérarchique auquel on souhaite se placer) grâce à des algorithmes de forces modifiés pour tenir compte de la taille du sommet (c'est-à-dire, de la taille de la communauté représentée). Là encore, tirant profit de la classification préalable des sommets, la représentation est extrêmement rapide (sa complexité dépendant du nombre de communautés et non plus du nombre de sommets du graphe).

Nous présentons dans la section 2 la méthode de classification hiérarchique proposée. La section 3 décrit la méthode de visualisation construite à partir de la classification hiérarchique. L'article se termine par la section 4 qui illustre l'approche en la comparant à des approches similaires sur plusieurs jeux de données réels dont un correspondant à un graphe biparti issu d'une grosse base de données d'actes notariés du Moyen-Âge.

2. Classification hiérarchique

La méthodologie présentée dans cet article vise à proposer une représentation synthétique et hiérarchique du graphe, afin de fournir à l'utilisateur une aide pour l'exploration, la compréhension et l'analyse de celui-ci. Le traitement du graphe s'effectue en deux étapes : la première consiste en une classification hiérarchique des sommets du graphe décrite dans la présente section.

Soumis au Journal de la Société Française de Statistique
File: rossi_villavialaneix_jsfds.tex, compiled with jsfds, version : 2009/12/09
date: 13 décembre 2011

2.1. Classification des sommets d'un graphe

Le but de la classification des sommets d'un graphe est la mise en valeur de sa structure macroscopique. En effet, la classification extrait des groupes de sommets, communément appelés *communautés*, qui sont connectés de manière dense et partagent (comparativement) peu de liens avec les sommets des autres communautés [15]. L'intérêt de cette décomposition est que l'utilisateur, qui cherche à explorer et comprendre le réseau, peut se focaliser sur un nombre réduit d'objets (les communautés) et leurs liens, plutôt que sur l'ensemble des sommets. Il pourra ensuite, naturellement, aller étudier de manière plus fine certains détails du graphe à l'intérieur d'une ou de plusieurs communautés d'intérêt.

Comme indiqué dans l'introduction, l'utilisation d'une classification des sommets d'un graphe pour en réaliser une représentation est une technique classique. Les résultats des approches s'appuyant sur une classification sont bien entendu très dépendants de sa pertinence, en particulier dans le cas de représentations simplifiées qui masquent le graphe sous-jacent [4, 35]. Or, contrairement aux données multi-dimensionnelles « classiques », le graphe n'induit pas de manière automatique une métrique ou une distance naturelle entre sommets. Les méthodes de classification de sommets d'un graphe ont donc un développement plus récent ; elles sont basées sur la définition de mesures de similarité entre sommets (qui s'appuient parfois sur le plongement d'un graphe dans un espace euclidien), sur des méthodes génératives qui supposent que le graphe est généré à partir d'un modèle aléatoire dans lequel les densités intra et inter-communautés différent, comme dans [10, 42] ou bien encore, sur l'optimisation d'un critère de qualité de la classification, comme la populaire modularité introduite dans [26]. Les revues [15, 36] donnent un panorama très complet des méthodes de classification de sommets dans un graphe et des questions ouvertes que celle-ci peut encore susciter.

Nous proposons d'utiliser ici l'optimisation de la modularité, pour deux raisons : le succès pratique de cette mesure [15] et le lien observé dans [28] entre la maximisation de la modularité et les modèles de forces utilisés pour visualiser les graphes. Ce dernier aspect implique que la modularité est particulièrement adaptée pour les problèmes de visualisation et constitue une pré-étape pertinente pour représenter le graphe.

La modularité est une mesure de qualité de la classification qui compare la disposition des arêtes à l'intérieur des communautés à celle qui serait obtenue dans un modèle nul dans lequel les poids des arêtes ne dépendent que du degré des sommets qu'elles joignent. De manière plus précise, pour un graphe $\mathscr{G} = (V, E, W)$, de sommets $V = \{x_1, \ldots, x_n\}$, d'arêtes E pondérées par les poids $(W_{ij})_{ij}$ (avec $W_{ij} \geq 0$ et $W_{ij} = W_{ji}$) et une partition des sommets en K classes $\mathscr{C}_1$, …, $\mathscr{C}_K$, la modularité de la partition est égale à

$$\mathscr{Q} = \frac{1}{2m} \sum_{k=1}^{K} \sum_{i,j \in \mathscr{C}_K} \left(W_{ij} - \frac{d_i d_j}{2m} \right) \tag{1}$$

avec $d_i = \sum_{j \neq i} W_{ij}$ et $m = \frac{1}{2} \sum_i d_i$. Ainsi, une forte modularité est l'indication que les arêtes à l'intérieur des communautés sont plus nombreuses (ou ont un poids plus important) que dans le cas d'une répartition des arêtes qui serait indépendante de la structuration en communautés. Même si les auteurs de [16] ont montré que l'optimisation de cette mesure de qualité peut induire des problèmes de résolution (certaines petites communautés significatives peuvent ne

pas être détectées par optimisation de la modularité), celle-ci a néanmoins montré sa pertinence pour mettre en valeur la structure d'un réseau ; notamment, la comparaison avec un modèle nul dans lequel les poids des arêtes sont proportionnels aux degrés des sommets permet de moins pénaliser la classification dans deux classes distinctes de sommets de forts degrés, que certaines alternatives comme la minimisation du nombre d'arêtes entre les classes et les méthodes spectrales qui lui sont associées [40]. Ceci limite l'obtention de classifications constituées d'une "super classe" (contenant la majorité des sommets et notamment les plus connectés) et de petites classes contenant des sommets plus marginaux (et ce type de classification est en général peu pertinent pour mettre en valeur la structuration du réseau).

Cependant, la maximisation de la modularité est un problème d'optimisation discrète NP-complet et nécessite donc un algorithme de résolution heuristique. Le meilleur compromis entre temps de calcul (qui permet de traiter de très gros réseaux) et qualité de l'optimisation semble avoir été atteint par les algorithmes gloutons à raffinement hiérarchique décrit dans [29]. Le principe général de ces méthodes ressemble à celui de la classification hiérarchique ascendante. On part d'une partition triviale associant une classe à chaque sommet du graphe. Les classes sont fusionnées de façon gloutonne en choisissant la meilleure fusion au sens d'un critère adapté (par exemple l'accroissement de modularité). La procédure s'arrête quand plus aucune fusion n'est possible sans dégrader la modularité. On passe alors à la phase de raffinement qui change des sommets de classe de façon opportuniste en cherchant à augmenter la modularité. Ce processus est mise en œuvre à plusieurs niveaux de granularité (c'est-à-dire à plusieurs niveaux dans le dendrogramme de la phase de fusion) afin de permettre des déplacements de groupes de sommets au lieu de se limiter à des sommets individuels.

La méthodologie utilisée dans cet article, détaillée dans l'algorithme 1, utilise une variante des recommandations de [29], avec notamment un test de connexité inédit :

1. la phase de fusion gloutonne (lignes 4 à 14 de l'algorithme 1) s'appuie sur le critère de priorité dit de *significativité* défini par

$$\mathrm{Sig}(\mathscr{C}_i,\mathscr{C}_j) = \frac{\Delta\mathscr{Q}_{\mathscr{C}_i,\mathscr{C}_j}}{\sqrt{\deg(\mathscr{C}_i)\deg(\mathscr{C}_j)}} \tag{2}$$

   où $\Delta\mathscr{Q}_{\mathscr{C}_i,\mathscr{C}_j}$ correspond à l'augmentation de modularité induite par la fusion des classes $\mathscr{C}_i$ et $\mathscr{C}_j$ et $\deg(\mathscr{C}_i)$ est la somme des poids des arêtes entre sommets de la classe $\mathscr{C}_i$. Comme indiqué dans [29], cette mesure de priorité permet de réaliser un compromis entre augmentation de la modularité et densité intra-classes ; elle est motivée par ses relations avec le modèle nul utilisé comme base de comparaison dans la mesure de la modularité. Nous utilisons une approche complète au sens où l'algorithme considère toutes les fusions possibles à chaque étape, plutôt que de se limiter par exemple à toutes les fusions d'une classe donnée avec les autres.

2. pour permettre un raffinement hiérarchique, les résultats intermédiaires de la phase de classification sont enregistrés (lignes 8 à 10 de l'algorithme 1) : de manière plus précise, les partitions de niveau 1, 2, ..., L, $(\mathscr{P}^l)_{1\leq l\leq L}$; sont enregistrées de telle sorte que la partition de niveau $l+1$ corresponde à une réduction d'au moins 25% du nombre de classes par rapport au niveau l. Ce choix correspond à un compromis entre la richesse des groupes de sommets considérés et le coût de la phase de raffinement. Notons que le niveau 1 correspond à la

partition triviale d'origine et que le niveau L est celui de la partition finale (pour laquelle aucune fusion n'est possible sans dégrader la modularité).

3. la procédure de raffinement multi-niveaux est ensuite mise en œuvre (lignes 16 à 30 de l'algorithme 1). Tout d'abord, à chaque niveau de classification $1 < l < L$, le graphe induit, $\mathscr{G}^l$ est défini : les sommets de $\mathscr{G}^l$ sont les classes du niveau l et les poids des arêtes entre sommets x_i^l et x_j^l dans $\mathscr{G}^l$ sont égaux à la somme des poids des arêtes du graphe initial entre les sommets affectés aux deux classes correspondant à x_i^l et x_j^l. On note $\mathscr{G}^1$ le graphe d'origine.

   Le raffinement multi-niveaux est alors effectué de manière descendante depuis le niveau $L-1$ jusqu'au graphe d'origine. Pour un niveau l donné (lignes 17 à 29), on cherche pour chaque sommet du graphe $\mathscr{G}^l$ le meilleur changement de classe possible dans la partition finale (de niveau L) au sens de la modularité. Chaque changement est mis en œuvre de façon gloutonne jusqu'à l'épuisement des possibilités d'amélioration. On passe alors au niveau inférieur $l-1$.

4. un contrôle de la connexité de chaque classe est enfin effectué (lignes 32 à 34 de l'algorithme 1) : si des classes se révèlent non connexes, elles sont scindées en autant de composantes connexes. Cette phase est à notre connaissance originale. Elle est indispensable dans notre contexte pour préserver la cohérence entre les différents niveaux de visualisation.

Enfin, éventuellement, la procédure en trois étapes précédemment décrite est partiellement répétée. En effet, le raffinement et le contrôle de connexité peuvent conduire à de nouvelles classes au niveau L pour lesquelles une fusion est avantageuse en terme de modularité. On réalise donc les opérations suivantes :

1. l'algorithme procède à une fusion des classes du graphe de niveau L par ordre de priorité si la modularité peut être augmentée de cette manière. Ceci conduit éventuellement à une partition de niveau $L+1$ (voire à une hiérarchie complète) ;
2. l'étape de raffinement multi-niveaux est appliquée en considérant, le graphe d'origine de niveau $l = 1$, le graphe de niveau L (celui de plus haut niveau de l'itération précédente) et la hiérarchie de fusions produite par l'étape précédente (en général réduite à une seule partition de niveau $L+1$) ;
3. le contrôle de la connexité de chaque classe est mis en œuvre comme précédemment.

En pratique, comme le montre l'algorithme 1 (ligne 34), il suffit d'appliquer de nouveau la procédure en trois étapes au graphe induit par $\mathscr{P}^L$ en considérant cette partition comme celle de niveau 2.

Ces étapes sont répétées jusqu'à stabilisation de l'algorithme. L'utilisation de cette procédure donne des résultats très compétitifs en terme de niveau de la modularité et permet de traiter des graphes de plusieurs milliers de sommets (plus de 10 000 dans certains exemples présentés dans la section 4) en quelques secondes sur un ordinateur de bureau.

2.2. Significativité d'une partition

L'un des principaux défauts des algorithmes de maximisation de la modularité, et en fait de la mesure elle-même, est qu'ils découvrent généralement une partition des sommets d'un graphe

Algorithme 1 Algorithme global d'optimisation de la modularité

Données : un graphe $\mathscr{G} = (V, E, W)$
Résultat : une partition $\mathscr{P} = (\mathscr{C}_i)_{1 \leq i \leq K}$ des sommets V du graphe de modularité maximale

$\mathscr{P}^1 = (\mathscr{C}_i^1 = \{x_i\})_{1 \leq i \leq |V|}$ {Partition initiale de niveau 1}
$L \leftarrow 2$; $\mathscr{P}^2 \leftarrow \mathscr{P}^1$
loop
  loop {boucle de fusion}
    $(i, j) \leftarrow \arg\max_{i \neq j} \mathrm{Sig}(\mathscr{C}_i^L, \mathscr{C}_j^L)$
    if la fusion des classes $\mathscr{C}_i^L$ et $\mathscr{C}_j^L$ augmente la modularité de $\mathscr{P}^L$ **then**
      fusionner les classes dans $\mathscr{P}^L$
      if $\frac{|\mathscr{P}^L|}{|\mathscr{P}^{L-1}|} < 0.75$ **then** {Sauvegarde de la partition}
        $L \leftarrow L + 1$; $\mathscr{P}^L \leftarrow \mathscr{P}^{L-1}$
      end if
    else
      terminer la boucle de fusion
    end if
  end loop
  if $L > 2$ **or** $\mathscr{P}^1 \neq \mathscr{P}^L$ **then** {si on a fusionné, on tente de raffiner la partition}
    for $level = L - 1$ **to** 1 **do** {raffinement multi-niveaux}
      $\mathscr{G}^{level} \leftarrow$ graphe induit par $\mathscr{P}^{level}$
      $\mathscr{Q}^{level} \leftarrow$ partition induite par $\mathscr{P}^L$ sur les sommets de $\mathscr{G}^{level}$
      loop {raffinement au niveau *level*}
        for all x sommet de $\mathscr{G}^{level}$ **do**
          for $k = 1$ **to** $|\mathscr{Q}^{level}|$ **do**
            calculer la modification de modularité induite par le passage de x dans la classe k de $\mathscr{Q}^{level}$
          end for
          mettre en œuvre l'échange de classes qui augmente le plus la modularité, si une augmentation est possible (dans $\mathscr{Q}^{level}$ et dans $\mathscr{P}^L$)
        end for
        if $\mathscr{P}^L$ n'a pas changé **then**
          terminer le raffinement au niveau *level*
        end if
      end loop
    end for
    if le raffinement a modifié $\mathscr{P}^L$ **then** {contrôle de connexité}
      $\mathscr{Q} \leftarrow \mathscr{P}^L$
      scinder dans $\mathscr{Q}$ les classes en composantes connexes
      $\mathscr{P}^2 \leftarrow \mathscr{Q}$; $L \leftarrow 2$
    else {aucune modification dans le raffinement}
      return $\mathscr{P}^L$
    end if
  else {la fusion a été sans effet, on ne peut plus améliorer la modularité}
    return $\mathscr{P}^L$
  end if
end loop

Soumis au Journal de la Société Française de Statistique
File: rossi_villavialaneix_jsfds.tex, compiled with jsfds, version : 2009/12/09
date: 13 décembre 2011

même si aucune partition significative n'existe. Ce phénomène s'explique par une forme de sur-apprentissage. En effet, la modularité augmente dès qu'on place dans une même classe deux sommets i et j connectés de façon significative, c'est-à-dire tels que $W_{ij} - \frac{d_i d_j}{2m} > 0$ (cf l'équation (1)). Il suffit donc de repérer de telles paires de sommets pour obtenir une partition de modularité positive, même si le graphe étudié ne contient pas de véritables communautés. Notons que des phénomènes similaires s'observent en général dans tout algorithme de classification.

Pour limiter ce problème, nous proposons d'estimer la distribution du maximum de la modularité sur un ensemble des graphes similaires au graphe étudié au niveau macroscopique : l'objectif est de mesurer la rareté de la partition obtenue en terme de niveau de modularité dans un ensemble de graphes similaires. Nous considérons le modèle dit de configuration [24] : il s'agit d'une distribution uniforme sur l'ensemble des graphes simples dont les sommets ont des degrés fixés. Pour un graphe donné, on considère donc l'ensemble des graphes avec le même nombre de sommets, dont lesdits sommets ont exactement les mêmes degrés que dans le graphe considéré, et qui ne possèdent ni boucle (arête d'un noeud vers lui-même), ni arête multiple.

Ce contexte a été étudié notamment dans [33] où les auteurs proposent une approximation de l'espérance du maximum de la modularité pour différents modèles de graphes aléatoires. L'approximation proposée n'est valable qu'asymptotiquement et pour des graphes denses, ce qui limite le potentiel applicatif du résultat. En outre, la dispersion du la modularité autour de son espérance n'est pas étudiée. Pour lever ces limites, nous estimons empiriquement la distribution du maximum de la modularité.

Pour ce faire, nous engendrons des graphes aléatoires uniformément dans l'ensemble choisi en utilisant une variante de l'algorithme simple MCMC proposé dans [34] : partant du graphe étudié, nous choisissons aléatoirement uniformément deux arêtes et nous échangeons leurs sommets, à condition que le graphe engendré ainsi ne comporte ni boucle ni arêtes multiples (les arêtes $A \leftrightarrow B$ et $C \leftrightarrow D$ deviennent ainsi $A \leftrightarrow D$ et $C \leftrightarrow B$). Après $Q|E|$ échanges ($|E|$ désigne le nombre d'arêtes du graphe), la chaîne de Markov ainsi engendrée est considérée comme stationnaire, ce qui permet de conserver le graphe obtenu (cf l'algorithme 2).

Algorithme 2 Algorithme pour engendrer un graphe aléatoire

Données : un graphe simple $\mathcal{G} = (V, E)$
Résultat : un graphe aléatoire simple dont les sommets ont exactement les mêmes degrés que ceux de $\mathcal{G}$

```
G_a = (V, E_a) ← G
for l = 1 to Q|E| do
    choisir (A, B) ∈ E_a aléatoirement uniformément
    choisir (C, D) ∈ E_a aléatoirement uniformément
    construire G'_a = (V, E'_a) en posant
        E'_a = (E_a \ {(A,B),(C,D)}) ∪ {(A,D),(C,B)}
    if G'_a = (V, E'_a) est simple then
        G_a ← G'_a
    end if
end for
return G_a
```

Nous suivons les recommandations de [23] en fixant Q à 100. Notons que cette valeur conduit à un processus relativement coûteux en temps calcul sur de gros graphes. Cependant, le coût reste

linéaire en $|E|$ pour tout graphe, comme pour les méthodes alternatives les plus efficaces. Par exemple, l'algorithme exact le plus efficace connu actuellement est en $|E|d_{\max}$ si $d_{\max}$ désigne le degré maximal des sommets du graphe, mais n'est applicable que si $d_{\max} = \mathcal{O}(|E|^{\frac{1}{4}-\tau})$ pour un $\tau > 0$ [3]. Cette hypothèse est très restrictive pour les graphes réels dont le degré maximal tend à augmenter aussi vite que le nombre d'arêtes. Par exemple dans le cas d'une distribution des degrés suivant une loi puissance d'exposant $\beta > 2$, on montre que le nombre d'arêtes et le degré maximum sont en $\Omega(|V|)$ [6], ce qui est incompatible avec l'hypothèse.

Le fait que l'algorithme utilisé produit asymptotiquement des graphes aléatoires distribués uniformément dans l'espace considéré n'est pas immédiat car la procédure est différente de celle de [34]. Comme indiqué dans [32], la chaîne de Markov construite par ce type de méthode n'est pas toujours irréductible. Quand on considère en effet des graphes orientés, on ne peut pas passer du graphe $A \to B \to C \to A$ au graphe obtenu en inversant les liens ($A \to C \to B \to A$) par une série d'échanges comme ceux utilisés dans l'algorithme retenu. Pour pallier ce problème, on combine donc les permutations entre 4 sommets utilisées par l'algorithme 2 avec l'opération d'échange entre les deux cycles orientés de trois sommets qui viennent d'être décrits. Le théorème 2 de [32] garantit alors qu'une chaîne de Markov combinant les deux types d'échange est irréductible. Dans notre cas, l'opération sur trois sommets est inutile car les graphes considérés ne sont pas orientés (l'échange ne modifie donc pas le graphe). De ce fait, le théorème 2 de [32] garantit que la chaîne de Markov produite par l'algorithme 2 est irréductible. D'autre part, la probabilité de rejet du graphe $\mathcal{G}'_a = (V, E'_a)$ n'est jamais nulle car elle est minorée par $\frac{1}{|E_a|^2}$ qui est la probabilité de choisir deux fois la même arête (A,B) (ce qui conduit à la création d'une arête double). La chaîne de Markov est donc apériodique. Enfin, la loi stationnaire de la chaîne est uniforme car les probabilités de transition entre deux graphes sont symétriques. En effet, bien que le choix de la permutation soit différent de celui utilisé dans [34], l'argument de symétrie de cet article s'applique ici : quand la transition d'un graphe $\mathcal{G}_a$ à un graphe $\mathcal{G}'_a$ est possible, elle se fait avec la probabilité de choisir les deux arêtes dont la permutation assure la transformation, soit $\frac{1}{|E_a|^2}$. Comme $|E|$ est constant dans l'ensemble des graphes considérés, cette probabilité est constante et donc symétrique.

Pour chaque graphe aléatoire ainsi obtenu, nous appliquons l'algorithme de partitionnement par maximisation de la modularité décrit dans la section précédente. Nous obtenons ainsi une estimation de la distribution de la modularité sur l'ensemble des graphes de même distribution des degrés que le graphe considéré (notons que si le graphe d'origine est biparti, le mécanisme de simulation choisi peut être modifié pour conserver cette structure). Cette distribution permet d'estimer la *p-value* pour la partition obtenue sur le graphe d'origine de la statistique de la modularité maximale [1] sous l'hypothèse nulle que les graphes étudiés sont choisis uniformément dans l'ensemble des graphes de distribution de degrés fixée. Selon cette hypothèse, la distribution des arêtes dépend uniquement de la distribution des degrés et pas d'une éventuelle structure en classes. Le rejet de l'hypothèse peut donc être vu comme un indice de l'existence d'une structure particulière de connexions. Si la modularité est élevée, cette structure correspond à une partition.

En pratique, nous construisons 100 graphes aléatoires et considérons la partition obtenue sur le graphe d'origine comme significative si sa modularité est supérieure aux 100 valeurs obtenues

[1] la statistique est elle-même calculée de façon approximative puisqu'on ne sait pas obtenir en un temps raisonnable le maximum exact de la modularité sur un graphe donné.

pour les graphes aléatoires, c'est-à-dire si la p-value est estimée inférieure à 1%.

2.3. Construction d'une hiérarchie

Une autre limitation de la modularité est sa tendance à trop favoriser les grosses classes au détriment des structures plus fines [16]. Pour limiter cet effet, nous construisons une classification hiérarchique du graphe. Celle-ci est obtenue en appliquant récursivement la méthode décrite en section 2.1. Plus précisément, nous construisons d'abord une partition optimale du graphe d'origine. Chaque classe de cette partition induit un sous-graphe du graphe d'origine, obtenu en ne conservant que les sommets inclus dans la classe et les arêtes qui les lient (les arêtes externes sont donc supprimées). On applique alors la méthode de classification dans chacun de ces sous-graphes, et ceci récursivement.

Deux critères sont utilisés pour terminer la récursion. Dans certaines situations, la meilleure partition découverte par l'algorithme est réduite à une classe unique, ce qui empêche naturellement la poursuite du processus. D'autre part, la technique de simulation proposée dans la section précédente est appliquée à chaque mise en œuvre de l'algorithme de classification. Si la meilleure partition obtenue pour un sous-graphe est jugée non significative, ce sous-graphe n'est pas partitionné mais au contraire conservé comme une classe unique. Ces deux critères conduisent en général à une classification hiérarchique déséquilibrée : certaines parties du graphe initial sont subdivisées finement, alors que d'autres restent dans une classe grossière déterminée dans la première phase de l'algorithme (cf par exemple la figure 8). Cette technique s'adapte donc à une densité non uniforme et à une structure complexe. L'algorithme complet d'obtention d'une classification hiérarchique est décrit dans l'algorithme 3.

Algorithme 3 Algorithme de construction d'une classification hiérarchique

Données : Un graphe $\mathscr{G} = (V, E, W)$
Résultat : Une classification hiérarchique $\mathscr{H}(\mathscr{G}) = (\mathscr{P}^l)_{1 \leq l \leq L}$
construire une partition $\mathscr{Q} = (\mathscr{C}_k)_{1 \leq k \leq K}$ de $\mathscr{G}$ par l'algorithme 1
if $K = 1$ **then** {classe unique}
return $(\mathscr{P}) = (\{V\})$ {le résultat est la hiérarchie triviale à un niveau}
else if la partition $\mathscr{Q}$ n'est pas significative (cf algorithme 2) **then**
return $(\mathscr{P}) = (\{V\})$ {hiérarchie triviale}
else
for $k = 1$ **to** K **do**
$\mathscr{G}_k \leftarrow$ sous-graphe de $\mathscr{G}$ dont les sommets sont dans $\mathscr{C}_k$
calcul de $\mathscr{H}(\mathscr{G}_k) = (\mathscr{P}_k^l)_{1 \leq l \leq L_k}$ par application récursive de l'algorithme 3
end for
$\mathscr{H}(\mathscr{G}) = (\mathscr{P}^l)_{1 \leq l \leq L}$ avec

$$\begin{aligned} L &= 1 + \max_{1 \leq k \leq K} \{L_k\} \\ \mathscr{P}^1 &= \mathscr{Q} \\ \mathscr{P}^{l+1} &= \left\{ \mathscr{P}_1^{\min(l, L_1)}, \ldots, \mathscr{P}_K^{\min(l, L_K)} \right\} \end{aligned}$$

return $\mathscr{H}(\mathscr{G})$
end if

Il est important de noter que la hiérarchie obtenue n'a en général que peu de rapport avec la

hiérarchie construite par la méthode décrite précédemment pour obtenir la partition optimale du graphe. En effet, le degré d'un sommet considéré dans un sous-graphe est réduit par rapport à son degré dans le graphe d'origine, ce qui modifie totalement la significativité de ses connections. De ce fait, une fusion de classes intéressantes dans le cas du graphe complet peut très bien devenir néfaste dans le cas d'un sous-graphe (et vice-versa).

La phase de représentation décrite dans la section suivante s'appuie sur un parcours de la hiérarchie : l'exploration commence naturellent au niveau le plus grossier (de modularité maximale). Les phases suivantes sont réalisées de façon gloutonne : à chaque étape de l'exploration, on remplace une classe par ses sous-classes, sans remise en question des étapes précédentes. La classe explorée est celle dont l'éclatement conduit à la réduction la plus faible de la modularité. On remplace donc la hiérarchie d'origine $\mathscr{H}(\mathscr{G}) = (\mathscr{P}^l)_{1\leq l\leq L}$ par une nouvelle hiérarchie $\mathscr{H}'(\mathscr{G}) = (\mathscr{P}'^l)_{1\leq l\leq L'}$ telle que le passage de la partition $\mathscr{P}'^l$ à la partition $\mathscr{P}'^{l+1}$ consiste uniquement en le remplacement d'une unique classe de $\mathscr{P}'^l$ par ses sous-classes.

Bien qu'il soit souvent possible de descendre au niveau le plus fin de la hiérarchie sans réduire la modularité en dessous du niveau de significativité obtenu par la méthode décrite dans la section 2.2, cela n'est pas nécessairement la meilleure solution. En effet, les représentations de graphe deviennent rapidement illisibles quand le nombre de sommets augmente. Nous nous sommes donc limités dans l'exploration à la partition la plus fine comportant moins de 100 classes. Ce critère simple combiné ainsi que l'exploration gloutonne de la hiérarchie, conduit à des représentations visuelles satisfaisantes (cf la section 4). On peut cependant imaginer d'autres stratégies, depuis une exploration classique par niveau (comme dans [1, 2, 37]) jusqu'à l'utilisation d'heuristiques d'optimisation combinatoire pour rechercher le meilleur développement pour un nombre de classes données (comme dans [30] par exemple). Ces questions importantes dépassent le cadre de cet article.

3. Représentation

Nous construisons des représentations simplifiées du graphe à partir de la classification hiérarchique obtenue. Le principe général de notre méthode consiste à utiliser la partition la plus grossière comme guide fondamental. En effet, cette partition maximise la modularité et est donc considérée comme la plus représentative de la structure du graphe. L'analyste peut ensuite demander des détails grâce à l'éclatement d'une classe en sous-classes. Nous respectons le principe de cohérence des *clustered graph visualizations* : le développement d'une classe en sous-classes ne modifie pas le rendu du reste du graphe des classes et les sous-classes sont représentées à proximité de l'emplacement qu'occupait la classe d'origine. Pour mettre en œuvre ce principe, il suffit donc de prévoir dans la représentation du graphe de la partition la plus grossière suffisamment de place pour procéder au développement des sous-classes. Nous estimons cette occupation des sous-classes et nous utilisons un algorithme de rendu capable de la prendre en compte dans le calcul de la représentation.

3.1. Estimation de l'occupation des sous-classes

Comme dans la plupart des méthodes de visualisation travaillant sur un graphe simplifié [4, 35], nous représentons les classes les plus fines par des disques dont la surface est égale à la racine

carrée du nombre de sommets regroupés dans la classe.

L'occupation des autres classes est estimée récursivement. Pour chaque classe regroupant d'autres classes, nous calculons une visualisation par l'algorithme décrit dans la section suivante en représentant chaque sous-classe par un disque de surface obtenue par estimation récursive. Nous calculons ensuite un disque englobant la visualisation ainsi obtenue, ce qui fournit l'estimation de la surface occupée par la classe considérée. La figure 1 illustre ce processus sur une hiérarchie simple à deux niveaux.

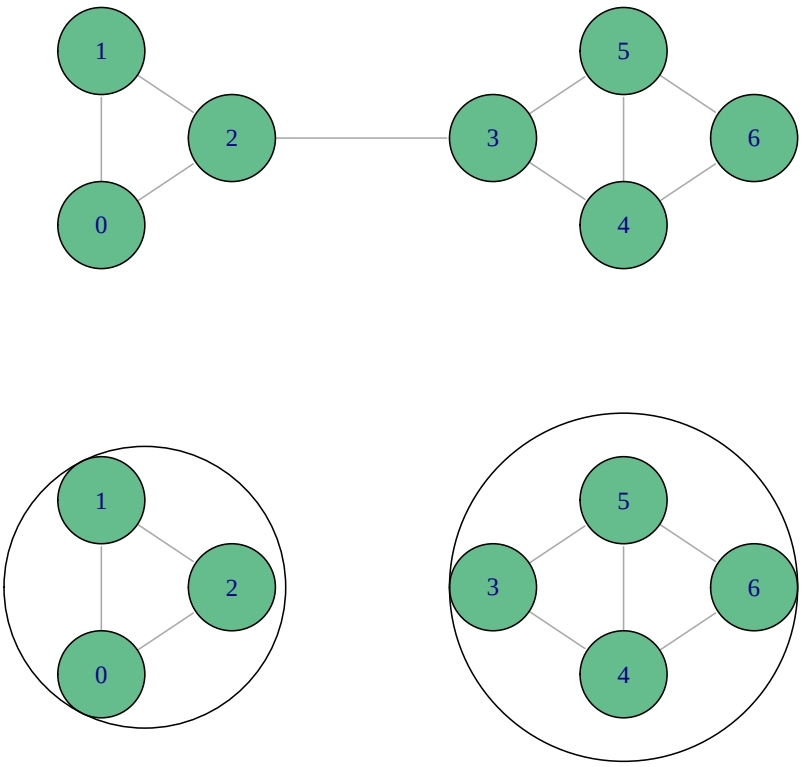


FIGURE 1. *Exemple d'estimation de l'occupation des classes : les sommets du graphe d'origine (figure du haut) sont partitionnés en deux classes,* $C0 = \{0,1,2\}$ *et* $C1 = \{3,4,5,6\}$ *dont les visualisations sont calculées indépendamment pour fournir une estimation d'occupation par cercles englobants (figure du bas)*

3.2. Modèle de forces

Pour calculer la visualisation d'un graphe, nous nous appuyons sur une des variantes de l'algorithme de Fruchterman Reingold [17] qui prennent en compte la taille des sommets [19, 39, 41]. Contrairement aux modèles de forces classiques, ces variantes utilisent un paradigme de ressort dont la longueur de repos n'est pas nulle [39] : deux sommets i et j connectés dans le graphe sont

reliés par un ressort qui exerce une force « attractive » d'intensité

$$f_a(i,j) = \delta_{W_{ij}>0}\,\text{signe}(\|x_i - x_j\| - r_i - r_j)\frac{(\|x_i - x_j\| - r_i - r_j)^2}{r_i + r_j + e}, \tag{3}$$

où x_i et x_j sont les positions des sommets dans la représentation graphique, r_i et r_j les rayons des disques représentant ces sommets et e un paramètre du rendu qui permet de régler les distances moyennes entre les sommets. Comme la force devient répulsive quand les sommets deviennent trop proches, ce modèle garantit en général que les disques associés aux sommets pourront être tracés sans superposition, à condition d'être associés à une variante adaptée des forces répulsives [39]. Deux sommets quelconques i et j se repoussent en effet selon une force d'intensité

$$f_{sr}(i,j) = \frac{(e + r_i + r_j)^2}{\|x_i - x_j\|}, \tag{4}$$

ou selon la variante suivante

$$f_{wr}(i,j) = \begin{cases} \frac{(e+r_i+r_j)^2}{\|x_i - x_j\|} & \text{si } \|x_i - x_j\| \leq e + r_i + r_j \\ \frac{(e+r_i+r_j)^3}{\|x_i - x_j\|^2} & \text{sinon.} \end{cases} \tag{5}$$

En pratique, ces choix assurent que la distance entre sommets à l'équilibre est supérieure à $r_i + r_j$, ce qui évite une superposition des disques associés.

Comme dans [39], nous passons progressivement des forces f_{sr} aux forces f_{wr} : les premières favorisent expérimentalement une bonne organisation globale du graphe, alors que les secondes réduisent la densité locale de la représentation, facilitant ainsi la réalisation de la contrainte de non superposition des disques associés aux sommets.

3.3. Visualisation complète

L'algorithme de visualisation complète procède de la façon suivante :

1. la surface occupée par chaque classe est estimée récursivement ;
2. une visualisation du graphe des classes les plus grossières est calculé selon le modèle de forces décrit dans la section 3.2, ce qui conduit, pour le graphe de la figure 1, à une représentation de la forme de celle de la figure 2 (qui montre l'absence de superposition des zones prévues pour le développement des classes en sous-classes) ;
3. on développe la classe qui induit la plus faible réduction de la modularité en calculant une visualisation du graphe de ses sous-classes :
   (a) le graphe des sous-classes est calculé ;
   (b) on lui ajoute des sommets virtuels [14] : chaque autre classe possédant au moins un sommet connecté à un sommet d'une des sous-classes est intégrée au graphe des sous-classes en tant que sommet virtuel ;
   (c) on calcule une visualisation du graphe des sous-classes en utilisant le modèle de la section 3.2 avec deux modifications : les sommets virtuels restent immobiles à

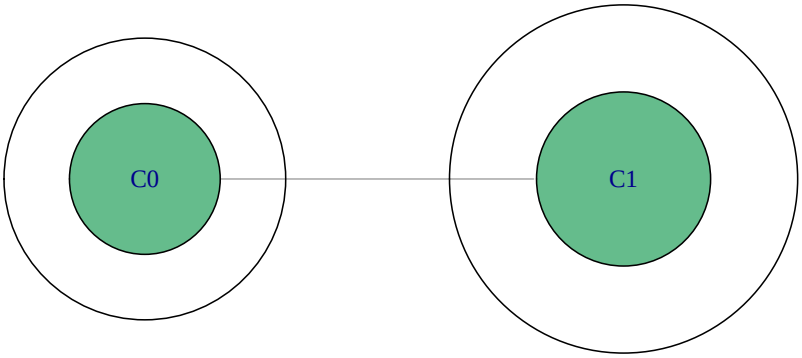


FIGURE 2. *Graphe des classes les plus grossières C0 et C1 pour le graphe de la figure 1 : les cercles entourant les disques matérialisent la surface nécessaire à la représentation des sous-classes*

l'emplacement que les classes correspondantes occupent dans le rendu actuel ; de plus, une force attractive centrée est incluse dans l'algorithme pour contraindre les sous-classes à rester autour de l'emplacement occupée par leur classe mère dans le rendu précédent.

Le mécanisme est illustré par la figure 3 qui montre le développement d'une classe pour le graphe grossier de la figure 2. La classe non développée ($C1$) forme un premier sommet virtuel (immobile). La classe mère ($C0$) est matérialisée par un autre sommet virtuel (aussi immobile) qui contraint les positions des sous-classes. On supprime ce sommet dans le rendu final.

3.4. Représentation des arêtes

La maximisation de la modularité conduit naturellement à un graphe de classes dont les arêtes ne sont pas significatives au sens de cette mesure. En effet, comme indiqué à la section 2.1, le poids de l'arête entre les classes $\mathscr{C}_i$ et $\mathscr{C}_j$ est donné par la somme des poids des arêtes entre les sommets affectés à $\mathscr{C}_i$ et à $\mathscr{C}_j$, soit donc

$$W_{\mathscr{C}_i,\mathscr{C}_j} = \sum_{a\in\mathscr{C}_i}\sum_{b\in\mathscr{C}_j} W_{ab}. \tag{6}$$

Si on applique cette formule pour la boucle reliant $\mathscr{C}_i$ à elle même, on obtient comme degré de la classe dans le graphe des classes la valeur suivante

$$d(\mathscr{C}_i) = \sum_{a,b\in\mathscr{C}_i} W_{ab}. \tag{7}$$

Une fusion des classes $\mathscr{C}_i$ et $\mathscr{C}_j$ conduit alors à une modification de modularité donnée par

$$\Delta\mathscr{Q}_{\mathscr{C}_i,\mathscr{C}_j} = \frac{1}{m}\left(W_{\mathscr{C}_i,\mathscr{C}_j} - \frac{d(\mathscr{C}_i)d(\mathscr{C}_j)}{2m}\right). \tag{8}$$

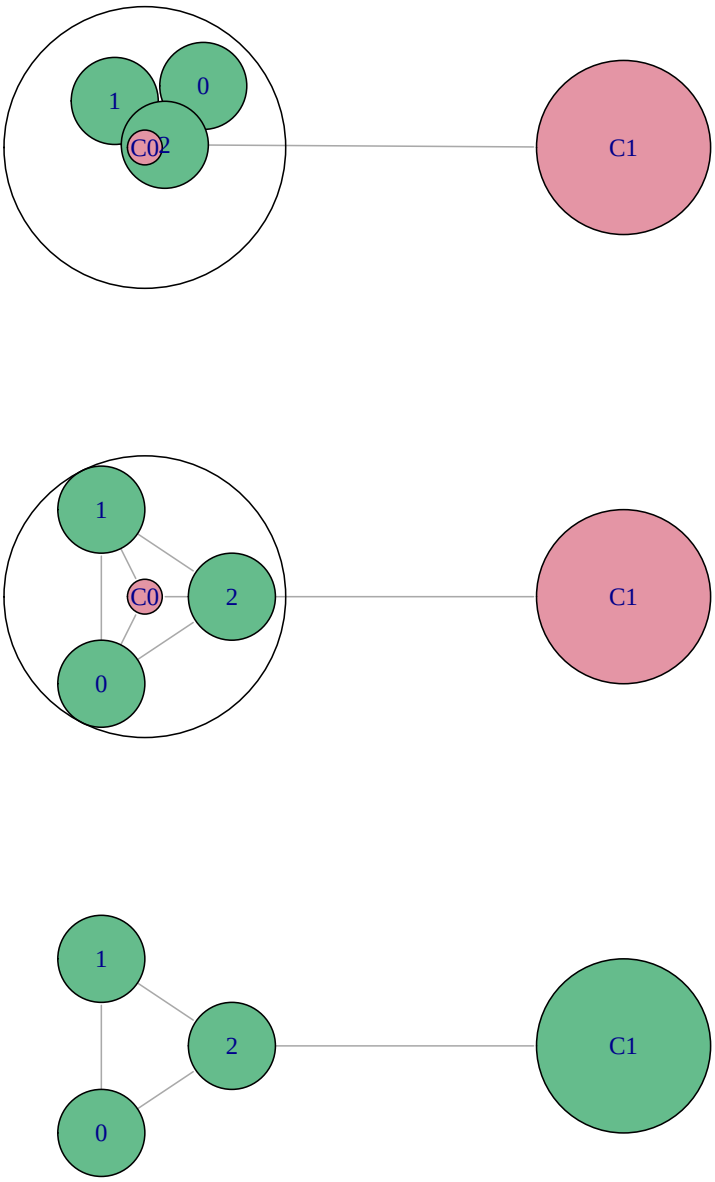


FIGURE 3. *Illustration du développement d'une classe en trois temps. La figure du haut correspond à une initialisation aléatoire de la position des sommets* $\{0,1,2\}$ *représentant les sous-classes de la classe* $C0$*, dans la zone d'occupation prévue pour cette classe. On constate la présence de la classe* $C1$ *comme sommet virtuel (immobile) en raison du lien entre le sommet* 2 *et un sommet de la classe* $C1$*. La classe* $C0$ *apparaît aussi comme sommet virtuel pour mettre en œuvre la force d'attraction centrée qui maintient les sommets* $\{0,1,2\}$ *dans la zone prévue pour* $C0$*. La figure centrale donne la position des sommets après optimisation du modèle de forces. Enfin, la figure du bas donne la représentation finale dans laquelle les sommets virtuels ont été soit supprimés (pour* $C0$*) soit remplacés par les sommets réels (*$C1$*).*

Soumis au Journal de la Société Française de Statistique
File: rossi_villavialaneix_jsfds.tex, compiled with jsfds, version : 2009/12/09
date: 13 décembre 2011

Comme la partition est optimale, cet accroissement est négatif, ce qui montre que l'intensité de la connexion $W_{\mathscr{C}_i,\mathscr{C}_j}$ n'est pas significative. Il est donc légitime de ne pas donner beaucoup d'importance dans la visualisation du graphe aux arêtes concernées.

Cependant, lors du processus de développement des classes, des arêtes significatives vont nécessairement apparaître : le raisonnement réalisé au dessus s'applique aussi mais l'accroissement de modularité est ici positif, puisque les sous-classes ont été fusionnées. Il est donc important de faire apparaître les arêtes concernées, et ce d'autant plus qu'elles sont significatives au sens de la mesure

$$W_{ij} - \frac{d_i d_j}{2m}. \tag{9}$$

Nous proposons donc d'utiliser un double codage pour représenter les arêtes à tout niveau d'exploration de la classification hiérarchique. La couleur des arêtes indique le signe de la mesure de significativité de l'équation (8) (bleu pour une valeur négative, rouge pour une valeur positive). L'épaisseur des arêtes est proportionnelle à leur significativité : les arêtes négatives ont une épaisseur de 1 et sont représentées en pointillés, alors que les arêtes positives sont en traits pleins dont l'épaisseur varient entre 1 et 5. La transformation linéaire de la significativité utilisée pour obtenir les épaisseurs des arêtes positives est calculée à partir des valeurs maximale prises par les significativités sur l'ensemble de la hiérarchie.

4. Exemples

Dans cette partie, nous proposons d'illustrer notre proposition sur plusieurs jeux de données : dans la section 4.1, nous comparons notre approche à d'autres méthodes de visualisation sur divers jeux de données publics. Ensuite, dans la section 4.2, nous appliquons notre méthodologie sur un jeu de données original issu d'un grand corpus de documents médiévaux qui représente un défi par sa taille.

Notre implémentation, réalisée en Java, est disponible publiquement [2]. Elle calcule les coordonnées des nœuds des graphes induits. Celles-ci peuvent ensuite être exploitées dans tout logiciel de visualisation de graphe qui permet l'importation de coordonnées pour les sommets. C'est le cas du *package* igraph [9] (destiné au logiciel R [31]) que nous avons utilisé pour réaliser les figures présentées dans la suite du texte.

4.1. Jeux de données publics

Nous proposons d'illustrer notre approche sur deux jeux de données publics, en la comparant, lorsque cela est possible, avec d'autres méthodes de visualisation, en particulier :

1. l'algorithme classique de Fruchterman et Reingold [17], appliqué directement au graphe entier. Nous nous sommes contentés ici d'une implémentation coûteuse en $O(|V|^2)$ par itération pour un graphe à $|V|$ sommets, ce qui limite la taille des graphes exploitables, d'autant qu'on considère généralement que l'algorithme nécessite $O(|V|)$ itérations pour converger. Il existe cependant des variantes plus efficaces (cf [39]), en $O(|E|+|V|\log|V|)$ par itération pour lesquelles le nombre d'itérations pour atteindre la convergence se comporte en pratique en $O(\sqrt{|V|})$. Ces coûts restent prohibitifs pour de très grands graphes ;

[2] `http://apiacoa.org/software/graph/index.en.html`

2. l'algorithme « LinLog » [17] [3], qui est basé sur l'optimisation d'une énergie dont le but est de mettre en valeur les communautés du graphe. L'algorithme utilise les optimisations évoquées au-dessus de [39]. Il est suffisamment rapide pour pouvoir traiter les plus gros graphes sur lesquels nous illustrons notre méthode (au maximum quelques dizaines de minutes sur un ordinateur de bureau) ;
3. l'optimisation de la modularité organisée, comme proposé dans [35]. Cette approche optimise, par recuit déterministe, un critère de qualité du plongement du graphe sur une grille de dimension 2 et de petite taille. Pour cette approche, nous fournissons, pour chacun des graphes traités, deux représentations :
   – une représentation simplifiée du graphe, induite par la classification. Dans celle-ci, chaque classe est représentée par un symbole (ici un disque) dont l'aire est proportionnelle au nombre de sommets de la classe. Les classes sont placées aux positions pré-définies de la grille et reliées par des arêtes dont l'épaisseur est proportionnelle à la somme des poids entre les sommets des deux classes ;
   – une représentation complète du graphe, obtenue par l'algorithme de Fruchterman et Reingold dans lequel chaque sommet a une position initiale correspondant au centre de la classe à laquelle il appartient. Cette représentation permet de traiter même de gros graphes grâce une pré-initialisation pertinente des sommets (dans les limites évoquées ci-dessus pour l'algorithme de Fruchterman et Reingold).
4. le modèle génératif de [21] : dans cette approche, la probabilité de connexion entre deux sommets du graphe est calculée à partir des positions de ces sommets dans un espace latent. L'estimation des paramètres de ce modèle est assez complexe et donc lente, ce qui le rend inapplicable sur de grands réseaux. Nous avons utilisé l'implémentation proposée par [22] dans l'ensemble de *packages* R statnet [18] pour illustrer l'approche sur les deux plus petits exemples présentés ici. Il faut noter que le modèle estimé par cette approche est très riche et n'est exploité ici que dans une de ses nombreuses dimensions (la visualisation), ce qui masque sa richesse et en donne une image artificiellement négative.

En complément, dans toutes les représentations, les sommets sont colorés :
– pour les représentations issues de notre méthode, les sommets et les classes sont colorés selon un code couleur correspondant à la classification du niveau le plus grossier (celui dont la modularité est optimale) ;
– pour les représentations issues de l'algorithme de Fruchterman et Reingold et de l'algorithme « LinLog » , la méthode de représentation du graphe n'induit pas une classification des sommets (le programme de l'algorithme « LinLog » fournit une classification mais elle est extérieure à l'optimisation de l'énergie et basée sur une optimisation de la modularité similaire à celle présentée dans la section 2). Nous utilisons donc, pour chaque sommet, le même code couleur que pour la représentation précédente pour faciliter la comparaison avec notre approche ;
– pour les représentations issues de l'algorithme d'optimisation de la modularité organisée, nous utilisons le code couleur induit par la classification sous-jacente à la méthode.

Enfin, pour les représentations simplifiées du graphe (celles produites par notre approche et celle produite par l'optimisation de la modularité organisée), nous représentons les arêtes entre

[3] disponible à `http://code.google.com/p/linloglayout/`

classes de sommets selon la méthode (épaisseur et code couleur) décrite dans la section 3.4. Pour la classification la plus grossière obtenue par notre approche, toutes les arêtes sont bleues car peu significatives du point de vue de la mesure de la modularité. Ce n'est pas forcément le cas pour l'approche consistant à optimiser la modularité organisée (et, en pratique, cela signifie que la fusion des deux classes reliées par une arête rouge ferait augmenter la modularité de la classification résultante).

4.1.1. Political books

En guise de premier exemple, nous utilisons un graphe de 105 livres politiques américains, tous publiés approximativement au moment de l'élection présidentielle de 2004 et qui étaient vendus en ligne par la société Amazon.com. Les sommets du graphe sont les livres et les 441 arêtes de ce graphe sont pondérées par le nombre de fois où les deux livres ont été achetés par le même acheteur. Ce réseau a été construit par Valdis Krebs et est disponible au téléchargement à l'adresse `http://www-personal.umich.edu/~mejn/netdata/polbooks.zip`. Une information supplémentaire est disponible sur les sommets (outre le titre du livre), son orientation politique (conservateur, libéral ou neutre), ce qui donne une méthode externe de validation de la pertinence des classes.

Le graphe complet est représenté dans la figure 4 : à gauche, l'algorithme de Fruchterman et Reingold fait apparaître les deux grands groupes de sommets correspondant aux deux orientations politiques principales (conservateur et libéral) des livres, mais l'impression de séparation est surtout induite par la densité des arêtes dans deux zones plutôt que par une séparation des groupes. À droite, l'algorithme « LinLog » permet effectivement de mieux séparer les deux groupes politiques de livres mais échoue à fournir une représentation agréable du graphe : les sommets sont fortement concentrés dans chacun des deux groupes, altérant la visibilité de l'ensemble du graphe. Ces tendances sont exacerbées sur une version non colorée des deux graphes, cf la figure 5.

Les résultats obtenus par le modèle génératif de [21] sont qualitativement assez comparables à ceux des autres méthodes (cf la figure 6). On observe cependant des différences intéressantes, en particulier une bonne séparation des différentes classes qui fait ressortir la relative rareté des connexions entre les deux principaux groupes de livres. On constate, comme pour les autres visualisations, que la suppression des couleurs rend les classes moins évidentes.

Une première représentation simplifiée du graphe est donnée par optimisation de la modularité organisée dans la figure 7 ainsi qu'une représentation obtenue en appliquant de manière modérée l'algorithme de Fruchterman et Reingold préalablement initialisé, pour tous les sommets, aux centres de leurs classes sur la grille. La représentation simplifiée fait apparaître deux grands groupes de sommets qui sont effectivement mis en valeur sur la représentation complète du graphe qui en est tirée. Cependant, bien que l'approche de modularité organisée de [35] conduise parfois à lever les limites de résolution de la modularité, ce n'est pas le cas sur cet exemple : on observe seulement quatre classes ce qui conduit à une représentation simplifiée très grossière. En outre, sa forme est contrainte de manière un peu artificielle par la grille sur laquelle la méthode est basée.

En revanche, la représentation obtenue par notre approche donne accès à quatre niveaux de détails pour des représentations simplifiées du graphe : la partition optimale (la plus grossière) permet de visualiser quatre grandes classes comme dans l'optimisation de la modularité organisée

FIGURE 4. *Graphe « Political books » représenté par l'algorithme de Fruchterman et Reingold (à gauche) et par l'algorithme « LinLog » (à droite). Les couleurs des sommets correspondent aux classes trouvées par optimisation de la modularité (comme décrit dans la section 2) et les étiquettes correspondent à l'orientation politique du livre.*

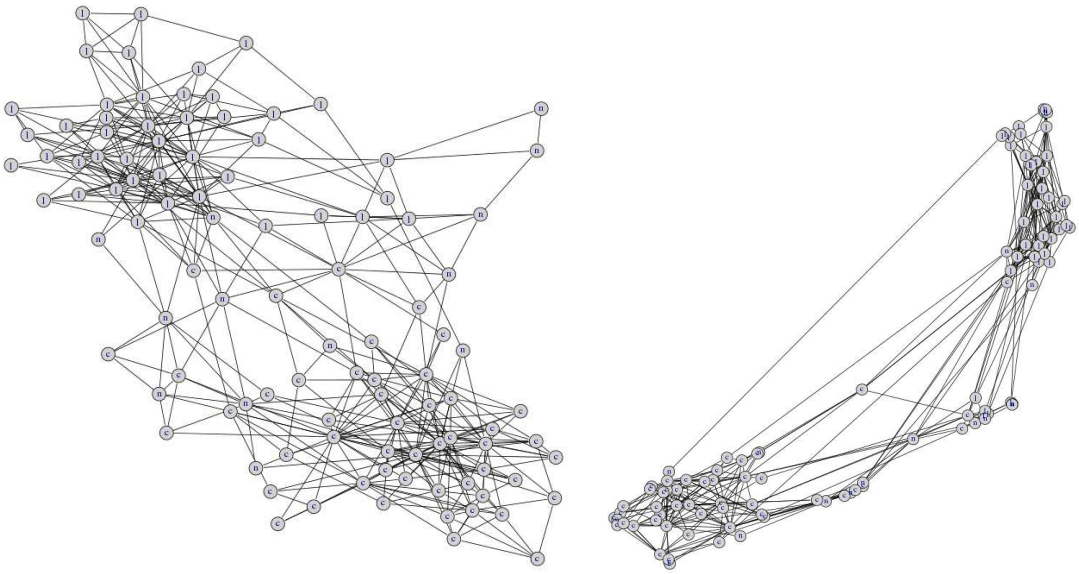


FIGURE 5. *Graphe « Political books » représenté par l'algorithme de Fruchterman et Reingold (à gauche) et par l'algorithme « LinLog » (à droite). Les couleurs ont été supprimées pour bien mettre en évidence les limites des deux approches.*

Soumis au Journal de la Société Française de Statistique
File: rossi_villavialaneix_jsfds.tex, compiled with jsfds, version : 2009/12/09
date: 13 décembre 2011

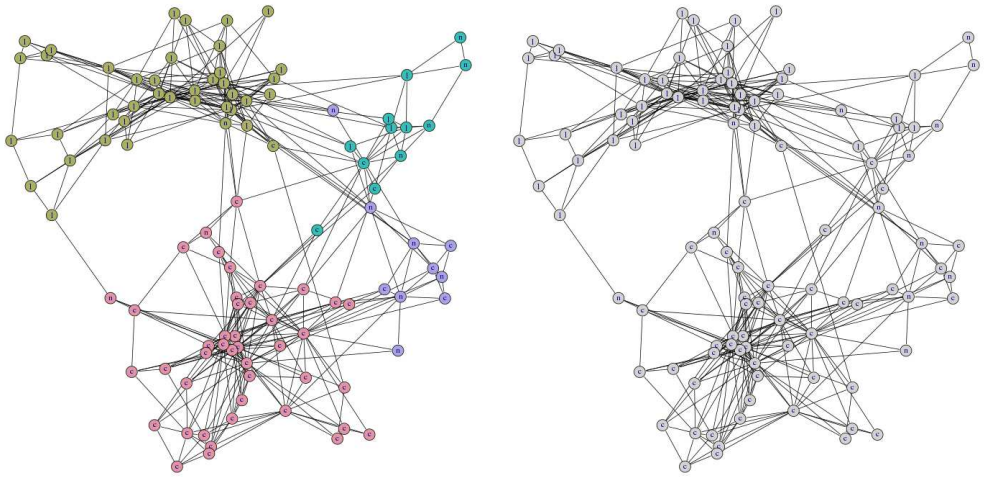


FIGURE 6. *Graphe « Political books » représenté par la méthode de [21], avec ou sans les couleurs correspondant aux classes trouvées par maximisation de la modularité.*

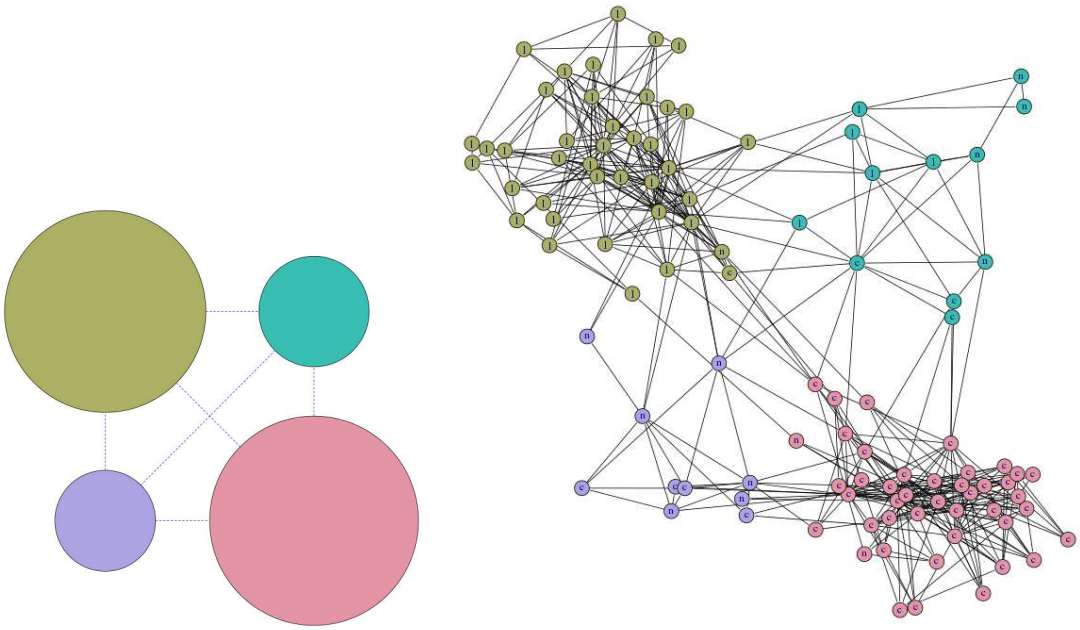


FIGURE 7. *Graphe « Political books » représenté par optimisation de la modularité organisée (à gauche : graphe induit par la classification et à droite : graphe complet obtenu par l'algorithme de Fruchterman et Reingold). Les couleurs des sommets correspondent aux classes trouvées par optimisation de la modularité organisée et les étiquettes correspondent à l'orientation politique du livre.*

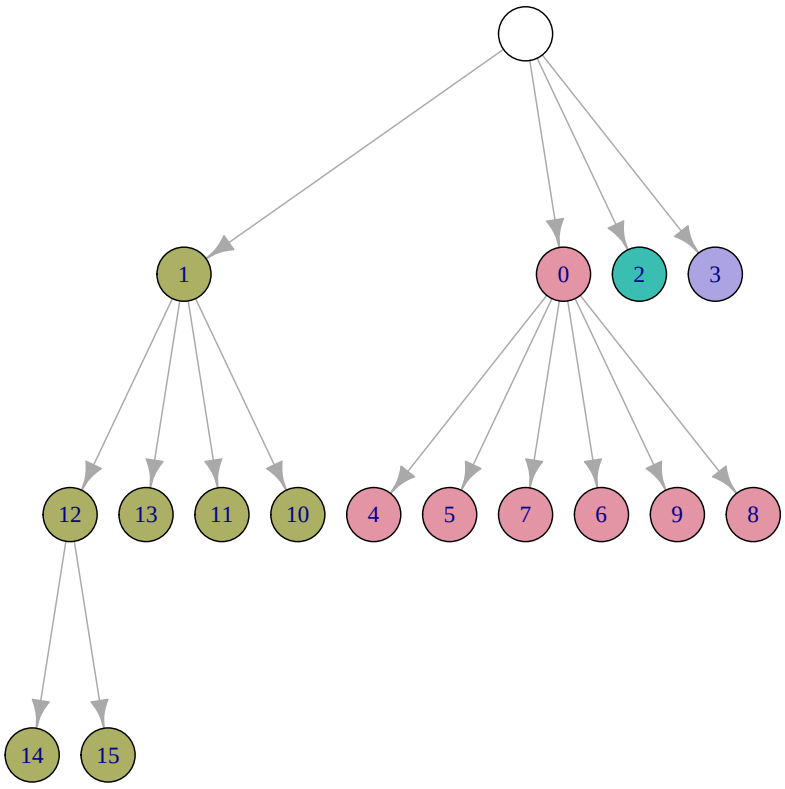


FIGURE 8. *Hiérarchie obtenue pour le graphe « Political books ».*

mais celles-ci sont placées de manière optimale par un algorithme de forces et non plus contraintes par la grille : cela permet de visualiser que les deux classes principales sont en fait les plus antagonistes alors que les deux plus petites classes ont une position plus centrale. La hiérarchie peut être raffinée trois fois : les deux plus grosses classes possèdent une sous-structure significative, qui peut être raffinée pour une sous-classe (cela correspond à une hiérarchie à trois niveaux représentée sur la figure 8). Ces trois raffinements, donnent accès à des détails de plus en plus fins sur la structure des deux grosses classes, permettant à l'utilisateur de visualiser dans ces classes-ci aussi des structures organisées qui ne sont visibles sur aucune des autres représentations. On remarque par exemple que la classe rose (à gauche) possède une structure plus complexe que la classe olive (à droite), sans que cela ne soit apparent sur les représentations globales.

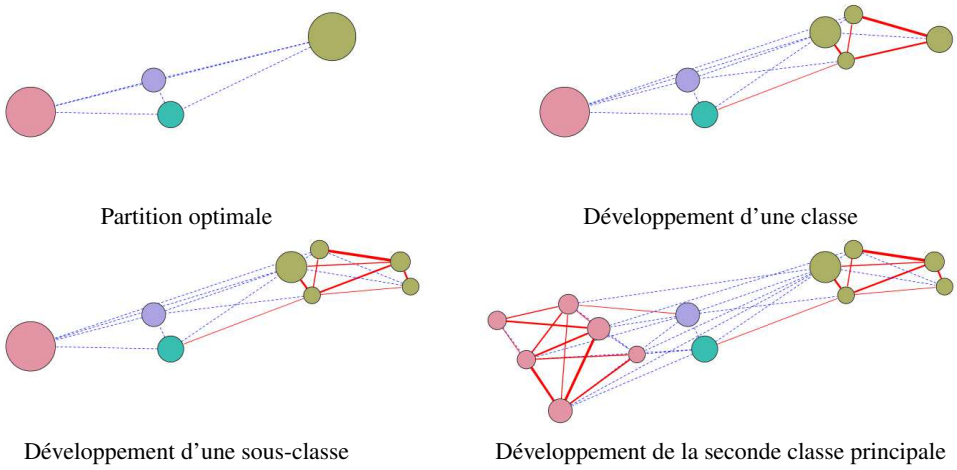


FIGURE 9. *Graphe « Political books » représenté par l'approche décrite dans l'article, avec une hiérarchie à trois niveaux. Les couleurs des sommets correspondent aux classes trouvées par optimisation de la modularité.*

La visualisation des arêtes met aussi en avant un phénomène intéressant : les arêtes significatives se retrouvent comme prévu majoritairement entre les sous-classes. On voit cependant que la classe rose n'est liée de façon significative qu'à la classe centrale violette, alors que la classe olive n'est liée significativement qu'à la classe centrale verte. Ces différences de connectivité n'apparaissent pas sur les autres représentations du graphe.

Notons pour conclure que les temps de calcul des visualisations sont très variables (mesurés sur un PC portable de type Core 2 duo à 2.4 Ghz). L'algorithme de Fruchterman et Reingold produit un résultat instantanément dans l'implémentation de igraph (ou de statnet). L'algorithme « LinLog » prend environ 4 secondes. Notre méthode construit la hiérarchie en 4 secondes et engendre les représentations graphiques en 2 secondes. En revanche, la méthode à variables latentes de [21] est très lente : le temps de calcul de la visualisation est de l'ordre 3 minutes, ce qui limite considérablement son domaine d'application. Cette tendance est confirmée dans la section suivante.

Soumis au Journal de la Société Française de Statistique
File: rossi_villavialaneix_jsfds.tex, compiled with jsfds, version : 2009/12/09
date: 13 décembre 2011

4.1.2. Netscience

Ce réseau modélise les collaborations entre scientifiques travaillant dans le domaine des réseaux. Il a été réalisé par Mark Newman en 2006 [25]. La plus grande composante connexe de ce réseau, sur laquelle nous travaillons, contient 379 sommets, représentant chacun un scientifique, et 914 arêtes, pondérées par le nombre de publications communes aux deux scientifiques.

Les représentations du graphe obtenues par l'algorithme de Fruchterman et Reingold et par l'algorithme « LinLog » sont données dans la figure 10. Les représentations du graphe obtenues par optimisation de la modularité organisée sont données dans la figure 11 (graphe simplifié et graphe entier) : dans ces deux représentations, les couleurs des sommets ne correspondent pas aux couleurs des sommets des autres représentations car les classes trouvées par optimisation de la modularité organisée ne sont pas les mêmes que les classes trouvées par optimisation de la modularité.

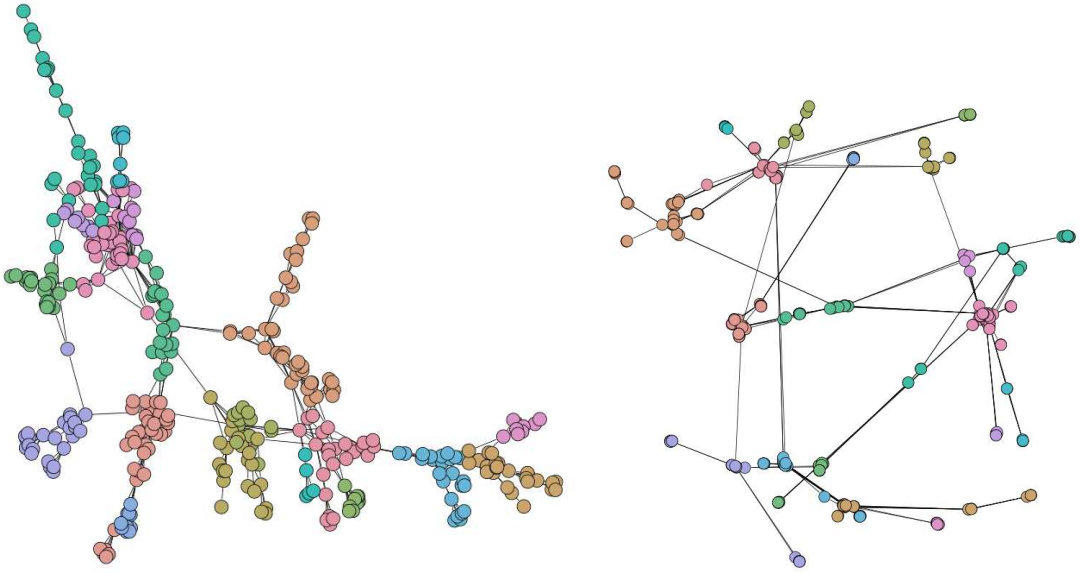


FIGURE 10. *Graphe « Netscience » représenté par l'algorithme de Fruchterman et Reingold (à gauche) et par l'algorithme « LinLog » (à droite). Les couleurs des sommets correspondent aux classes trouvées par optimisation de la modularité (comme décrit dans la section 2).*

Les mêmes phénomènes que ceux observés pour le graphe « polbooks » sont visibles : l'algorithme de Fruchterman et Reingold met en valeurs des groupes de sommets denses mais ils présentent deux inconvénients majeurs : le premier est que, à cause de la superposition des sommets, la figure est difficile à lire. Le second est que la séparation entre les groupes est insuffisamment mise en valeur : plusieurs groupes de sommets se superposent (par exemple, le groupe marron et le groupe bleu à gauche de la figure ou bien deux groupes dans les tons vert au centre de celle-ci). L'algorithme « LinLog » tend ici à trop fortement regrouper les sommets d'un même groupe : ils sont souvent pratiquement superposés sur la figure. Pour cette approche, l'optimisation de la modularité organisée échoue à fournir une simplification parlante du graphe

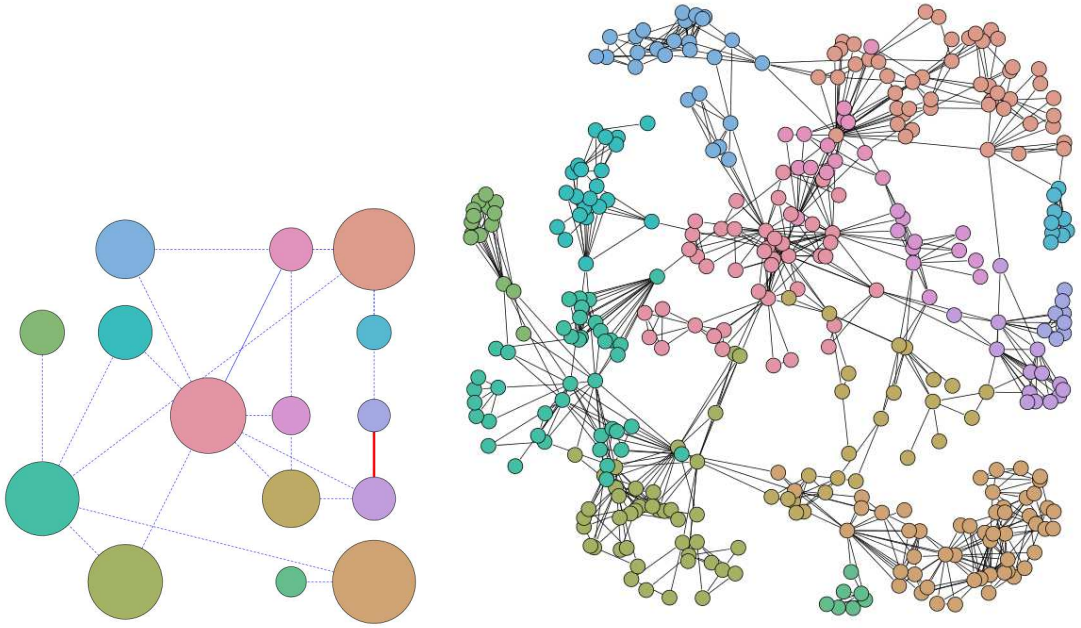


FIGURE 11. *Graphe « Netscience » représenté par optimisation de la modularité organisée (à gauche : graphe induit par la classification et à droite : graphe complet obtenu par l'algorithme de Fruchterman et Reingold). Les couleurs des sommets correspondent aux classes trouvées par optimisation de la modularité organisée.*

et, en conséquence, la représentation du graphe complet qui en est tirée est de meilleure qualité en ce qui concerne la limitation de la superposition des sommets que celle obtenue en appliquant directement l'algorithme de Fruchterman et Reingold mais elle est également d'une lisibilité très limitée à cause des superpositions d'arêtes.

Enfin, le modèle de Hoff et al. [21] donne des résultats intermédiaires entre l'algorithme de Fruchterman et Reingold et ceux de « LinLog » (cf la figure 12) : les sommets d'une même classe sont bien regroupés tout en restant visibles, ce qui limite les phénomènes de superposition des arêtes et rend le résultat global assez lisible comparativement aux trois autres méthodes. Malheureusement, le graphe « Netscience » comporte environ deux fois plus d'arêtes que le graphe « Polbooks » (et plus de trois fois plus de sommets). Cette augmentation de taille a un effet considérable sur le temps d'estimation des paramètres du modèle qui passe de 3 minutes sur « Polbooks » à 58 minutes sur « Netscience » . À titre de comparaison, notre approche construit la hiérarchie de classification en 4 secondes puis l'ensemble des représentations en 4 secondes, soit plus de 400 fois plus rapidement.

Notre approche fournit à l'utilisateur une classification hiérarchique en deux niveaux. L'exploration développement par développement de cette hiérarchie se fait en sept étapes, en épuisant ainsi tous les développements possible (il n'y a donc pas d'autres sous-structures dans ce graphe que celles fournies dans la dernière représentation). Dans la figure 13, nous présentons quatre étapes intéressantes. La représentation de plus forte modularité (« partition optimale ») donne une représentation simplifiée claire et compréhensible de la structure du graphe. Les représentations suivantes fournissent, progressivement, des détails de plus en plus fins sur des classes elles-mêmes

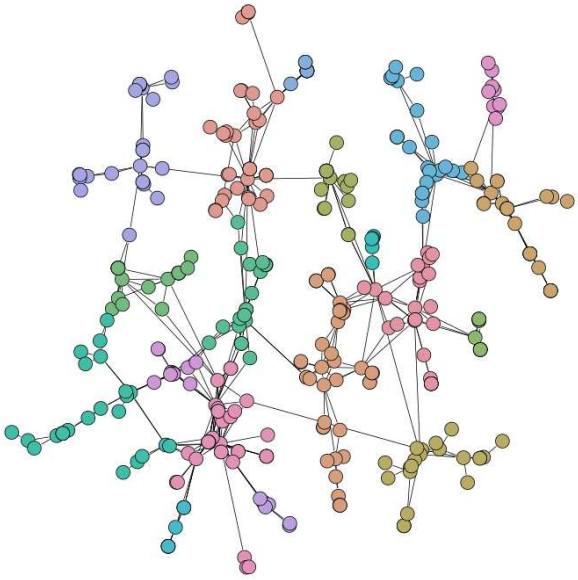


FIGURE 12. *Graphe « Netscience » représenté par la méthode de [21]. Les couleurs des sommets correspondent aux classes trouvées par optimisation de la modularité.*

en limitant la dégradation de la partition globale. Également, petit à petit, la représentation donne accès aux relations entre sous-groupes : il est ainsi possible de voir que certains sous-groupes d'une classe ne sont connectés au reste du graphe que par l'intermédiaire d'un groupe de sommets qui fait la liaison (c'est le cas, par exemple, de la classe violette en bas et au centre, développée en trois sous-classe). Cette structure localement arborescente se retrouve de façon moins claire dans les autres représentations : dans les représentations de la figure 13, on peut identifier clairement le groupe de sommets qui assure la connectivité d'un autre groupe de sommets au reste du graphe, alors que les autres figures indiquent seulement qu'un tel phénomène est probable. Sur cet aspect, la coloration des arêtes apporte une information supplémentaire : on peut alors constater sur le développement le plus fin que certaines arêtes entre sous-classes sont significatives. C'est le cas pour la classe verte centrale développée en dernier, ainsi que pour la classe la plus à gauche des figures, développée en quatre sous-classes.

4.2. Visualisation d'un grand graphe

Le graphe sur lequel nous illustrons notre approche est issu d'un grand corpus de documents médiévaux, homogènes au niveau de leur localisation géographique (seigneurie de Castelnau Montratier, Lot, France) et de leur type (ce sont tous des actes notariés et ils sont tous relatifs à des contractualisations de type « bail à fief »). Ces actes notariés ont été établis, principalement, entre le XIII$^{\text{ème}}$ et le XV$^{\text{ème}}$ siècles et constituent une collection proche de l'exhaustivité de ce type de contrats dans cette seigneurie pour la période entourant la guerre de cent ans. En effet, au XIX$^{\text{ème}}$ siècle, afin de pouvoir établir les droits féodaux du nouveau propriétaire de la seigneurie, un feudiste avait été mandé pour collecter et recopier ce type de contrats : c'est

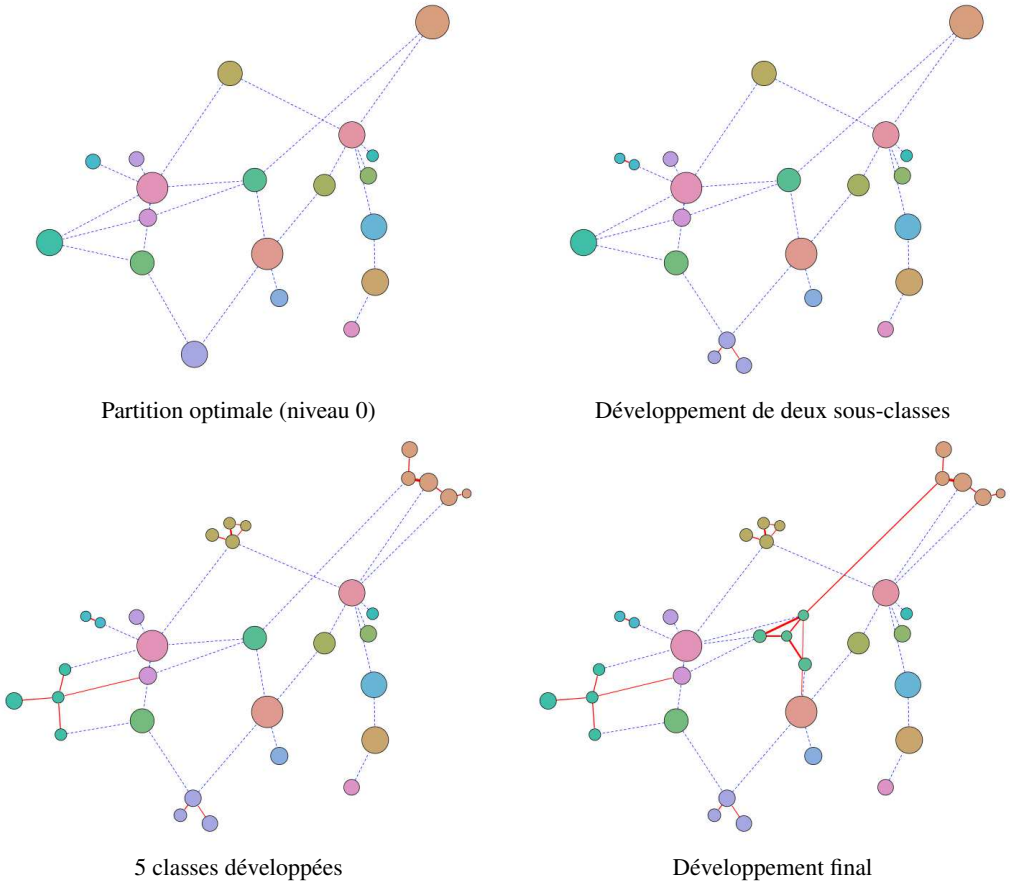


FIGURE 13. *Graphe « Netscience » représenté par l'approche décrite dans l'article, avec une hiérarchie à 2 niveaux : quatre étapes du développement sont représentés ici. Les couleurs des sommets correspondent aux classes trouvées par optimisation de la modularité.*

ce travail qui nous est parvenu. Chaque contrat notarié contient une ou plusieurs transactions qui impliquent des personnes qui sont des nobles ou de simples paysans (serfs ou paysan libre). Les transactions ont été modélisées et saisies dans une base de données disponible à l'adresse `http://graphcomp.univ-tlse2.fr/`. À partir de ces données, nous avons défini un graphe tel que :

- les sommets de ce graphe sont les transactions du corpus et les personnes actives de ces transactions (à l'exclusion, donc, des notaires et des témoins) ;
- un lien entre deux sommets modélise l'implication active d'une personne dans une transaction.

Le graphe est dit *biparti*, c'est-à-dire que ces sommets sont divisés en deux groupes (les transactions et les individus) avec aucune connexion entre sommets d'un même groupe. Le graphe final contient 3 918 individus impliqués dans 6 455 transactions (soit un total de 10 373 sommets).

La classification hiérarchique de ce graphe conduit à une hiérarchie à 3 niveaux, depuis une partition optimale à 48 classes jusqu'à une partition fine de 1 074 classes, après 70 étapes de raffinement. Comme indiqué précédemment, nous limitons l'exploration à moins de 100 classes, ce qui conduit dans ce cas à une partition en 93 classes, obtenue après seulement 6 étapes de raffinement. Ceci permet de limiter la complexité de la représentation présentée globalement à l'utilisateur.

La représentation simplifiée du graphe à partir de la classification du niveau le plus grossier est donnée par la figure 14 : celle-ci contient 48 classes composées de 10 à 740 sommets (216 sommets en moyenne par classe, avec une médiane égale à 171 sommets). Pour comparaison, la représentation du graphe complet, obtenue par application de l'algorithme « LinLog », est également donnée dans cette figure. Cette dernière représentation ne fait apparaître aucune sous-structure particulière et, à cause du grand nombre de sommets, n'est pas d'une grande aide pour l'exploration du graphe.

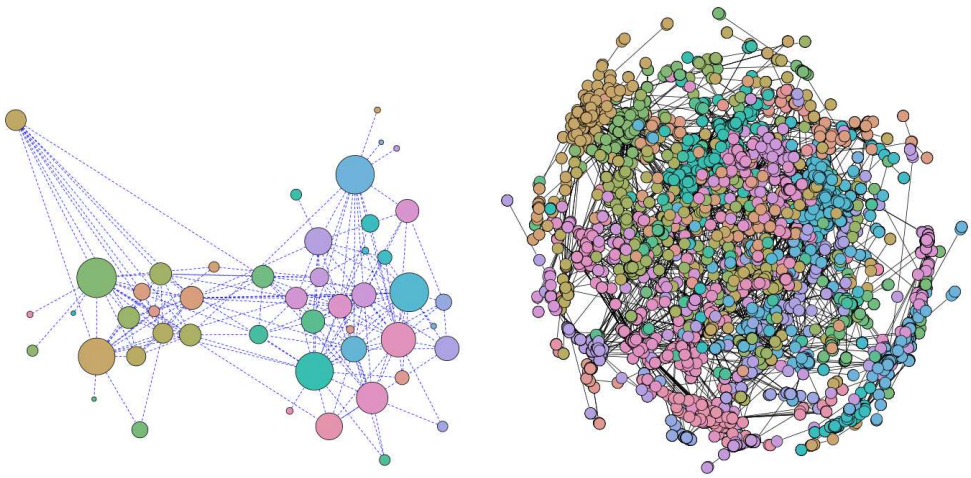


FIGURE 14. *Graphe issu du corpus d'actes notariés médiévaux, représenté de manière simplifiée par notre approche (à gauche) et par l'algorithme « LinLog » (à droite). Les couleurs des sommets correspondent aux classes trouvées par optimisation de la modularité organisée dans les deux représentations.*

Au contraire, la représentation du graphe obtenue par optimisation de la modularité est bien structurée et découpée en deux gros ensembles de classes. La figure 15 permet de visualiser les dates moyenne des transactions de chaque classe ainsi que l'écart type de celles-ci. En complément, la représentation issue de l'algorithme « LinLog » est également colorée de la même manière : les sommets correspondant à une transaction sont colorés selon le code couleur adopté pour la représentation de la moyenne des dates des classes issues de l'optimisation de la modularité et les sommets correspondant à un individu sont colorés selon le même code couleur pour une valeur correspondant à la moyenne des dates des transactions dans lesquelles ils apparaissent.

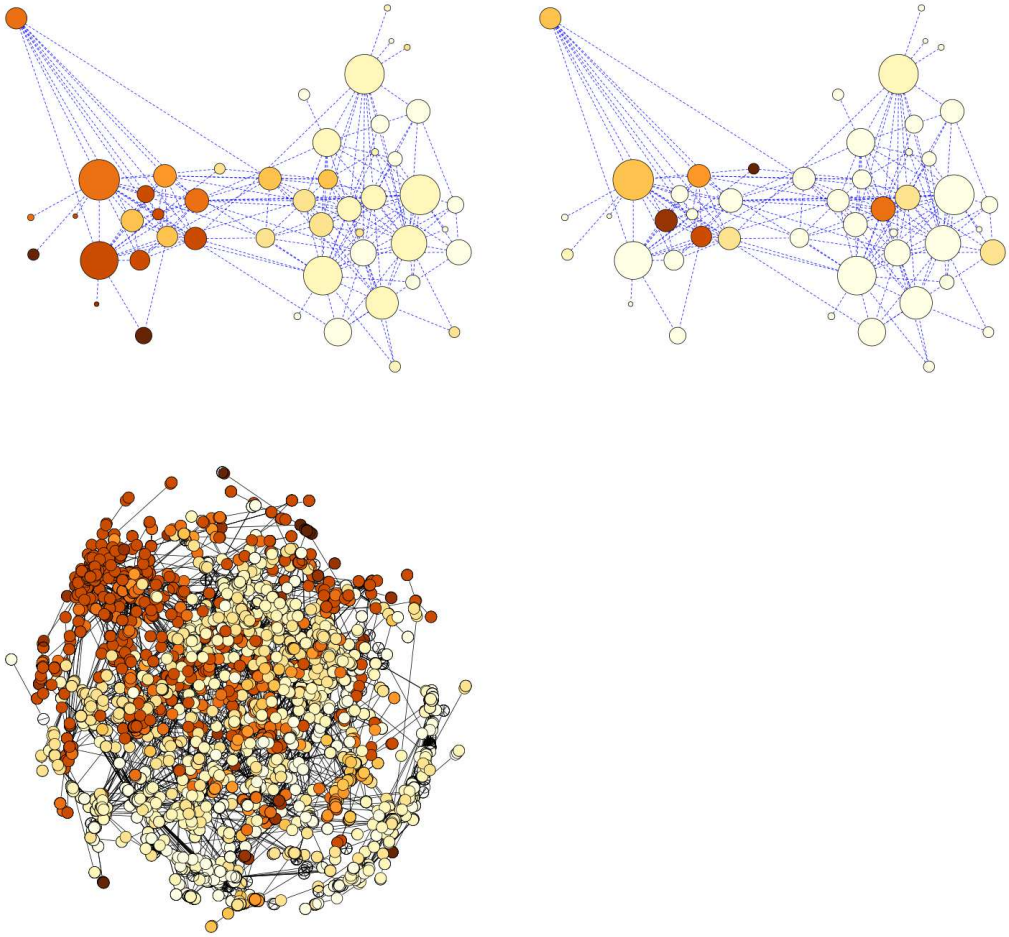


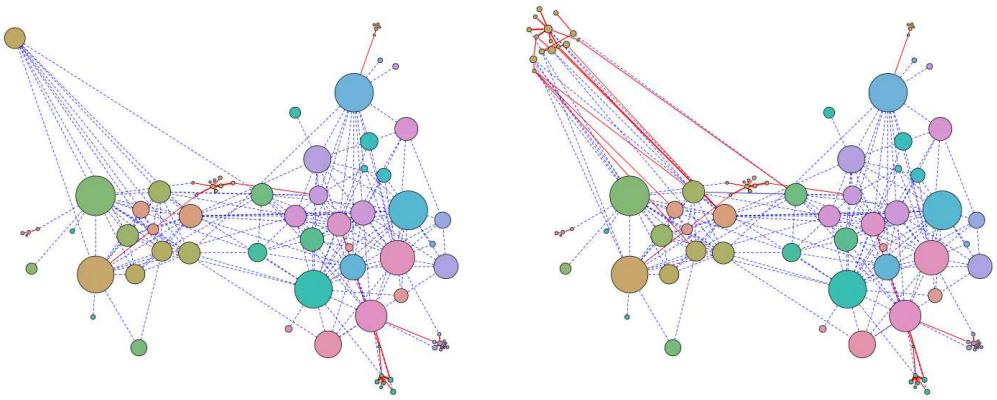
FIGURE 15. *Surimposition d'une information de datation sur les représentations du graphe issu du corpus d'actes notariés médiévaux : en haut à gauche, date moyenne des transactions de chaque classe ; en haut à droite, écart type des dates des transactions de chaque classe ; en bas : date (transactions) ou date moyenne d'activité (individus) des sommets. Les dates les plus anciennes correspondent aux couleurs les plus claires et les variabilités les plus faibles dans la valeur de la date au sein d'une classe correspondent aussi aux couleurs les plus claires.*

Soumis au Journal de la Société Française de Statistique
File: rossi_villavialaneix_jsfds.tex, compiled with jsfds, version : 2009/12/09
date: 13 décembre 2011

Sur cette figure, il apparaît clairement que les deux ensembles de classes identifiables sur la représentation issue de l'optimisation de la modularité correspondent en fait à deux ensembles de dates distinctes (en fait, avant et après la guerre de Cent Ans). La représentation issue de l'algorithme linlog fait bien, elle aussi, apparaître une organisation du graphe selon ce critère (la date) mais ne fait pas apparaître clairement la séparation franche entre ces deux parties du réseau : celle-ci est pourtant attestée historiquement par une forte diminution du nombre de transactions effectuées durant la période de la guerre de Cent Ans.

Deux versions plus détaillées de la hiérarchie de classifications obtenue à partir de l'approche décrite dans la section 2.3 sont donnés dans la figure 16. Les 5 premières étapes de développement de la hiérarchie correspondent en fait l'éclatement de petites classes et ne modifient pas fortement la représentation globale du graphe. On peut toutefois remarquer qu'une des classes de plus forte variabilité de dates (petite classe au centre de la figure, colorée en rouge foncée dans la représentation des écarts types des dates en haut à droite de la figure 15) fait partie des classes qui sont développées : la forte variabilité des dates de cette classe pourrait être expliquée par la sous-structure significative mise en évidence par notre approche. On remarque notamment que cette classe et ses sous-classes semblent jouer le rôle de pont entre les deux parties du graphe, à gauche et à droite, qui correspondent à deux périodes différentes.

En revanche, la sixième étape subdivise une classe située en périphérie à gauche de la représentation : cette classe est organisée en étoile autour d'un groupe central de sommets et sa représentation requiert un espace assez grand. C'est uniquement ce développement qui explique la position singulière de la classe sur les représentations de plus faible niveau (pour permettre sa représentation développée) et ceci est donc une limitation de notre approche due à une probable surestimation de la surface occupée par cette classe une fois développée. Cette limitation souligne paradoxalement l'importance de la classe en question dont les nombreuses sous-classes sont connectés de façon significative à plusieurs autres classes. Il semble donc important de diriger l'analyste vers le sous-graphe correspondant pour une première exploration.



5ème étape 6ème étape

FIGURE 16. *Représentation des 5ème et 6ème étapes du développement de la hiérarchie de classifications construite comme dans décrit dans la section 2.3 pour le graphe issu du corpus d'actes notariés médiévaux*

5. Conclusion

Nous avons présenté dans cet article une nouvelle approche pour l'exploration visuelle de grands graphes. Cette approche calcule une représentation résumée en deux dimensions d'un graphe en s'appuyant sur une classification de ses sommets obtenue par maximisation du critère de modularité. Pour permettre à l'analyste d'explorer le graphe sous-jacent de façon plus fine, la méthode proposée construit récursivement des classifications des sous-graphes associés à chaque classe, en se limitant aux structures significatives (grâce à l'estimation de la distribution de la mesure de modularité sur un ensemble de graphes aléatoires semblables au graphe étudié). La visualisation du graphe de la partition la plus grossière est calculée de manière à faciliter le raffinement de cette partition : toute classe qui possède une sous-structure significative peut être remplacée par le graphe des classes de cette sous-structure, en conservant la lisibilité de l'ensemble de la représentation.

La méthode proposée est rapide et ne nécessite aucun ajustement de paramètre. L'utilisateur doit choisir le niveau maximal de raffinement envisagé dans la visualisation, mais ce paramètre n'a pas d'influence sur le calcul de la hiérarchie : plusieurs visualisations peuvent être comparées en faisant varier ce paramètre, sans devoir recalculer la classification hiérarchique.

Les résultats obtenus sur trois graphes réels de tailles croissantes sont très satisfaisants. Ils montrent que la méthode réalise un bon compromis entre une représentation complète du graphe, qui est en général difficile à lire et à analyser, et une représentation unique de la partition de plus grande modularité qui peut être trop simplifiée en raison de la limite de résolution de la modularité. L'utilisation d'un critère de significativité est particulièrement important dans cette démarche car il limite fortement la tendance naturelle de l'analyste à étudier les détails fins du graphe sur une représentation complète. Ces résultats confirment ceux déjà obtenus avec une version antérieure et plus simple de la méthode, présentés dans [7, 8].

La principale limite de la méthode a été observée dans le cas du graphe médiéval : l'estimation récursive de la place occupée par une classe est parfois largement fausse, dans le sens d'une surestimation. Cela conduit en pratique à un graphe dont certaines arêtes sont très longues alors qu'elles semblent pouvoir être réduites de façon simple. Ce phénomène est une limitation connue des algorithmes de visualisation qui tiennent compte de l'occupation des sommets [19]. Nous étudions actuellement les méthodes proposées dans [19] pour homogénéiser les longueurs des arêtes dans ce contexte, ce qui permettra vraisemblablement de lever cette limite de notre méthode.

Remerciements

Les auteurs tiennent à remercier les deux rapporteurs pour leurs remarques qui ont permis d'étoffer et d'améliorer cet article.

Références

[1] D. AUBER, Y. CHIRICOTA, F. JOURDAN et G. MELANÇON : Multiscale visualization of small world networks. *In INFOVIS'03*, 2003.

[2] D. AUBER et F. JOURDAN : Interactive refinement of multi-scale network clusterings. *In International Conference on Information Visualisation, International Conference*, pages 703–709, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

Soumis au Journal de la Société Française de Statistique
File: rossi_villavialaneix_jsfds.tex, compiled with jsfds, version : 2009/12/09
date: 13 décembre 2011

[3] M. BAYATI, J. KIM et A. SABERI : A sequential algorithm for generating random graphs. *Algorithmica*, 58:860–910, 2010.

[4] R. BOULET, B. JOUVE, F. ROSSI et N. VILLA : Batch kernel SOM and related laplacian methods for social network analysis. *Neurocomputing*, 71(7-9):1257–1273, 2008.

[5] R. BOURQUI, D. AUBER et P. MARY : How to draw clustered weighted graphs using a multilevel force-directed graph drawing algorithm. *In Proceedings of the 11th International Conference Information Visualization, 2007. IV'07.*, pages 757–764, July 2007.

[6] F. CHUNG : Random graphs, a whirlwind tour of. *In* Robert A. MEYERS, éditeur : *Encyclopedia of Complexity and Systems Science*, pages 7493–7505. Springer New York, 2009.

[7] S. CLÉMENÇON, H. DE ARAZOZA, F. ROSSI et V.C. TRAN : Hierarchical clustering for graph visualization. *In Proceedings of XVIIIth European Symposium on Artificial Neural Networks (ESANN 2011)*, pages 227–232, Bruges, Belgique, April 2011.

[8] S. CLÉMENÇON, H. DE ARAZOZA, F. ROSSI et V.C. TRAN : Visual mining of epidemic networks. *In* Joan CABESTANY, Ignacio ROJAS et Gonzalo JOYA, éditeurs : *Advances in Computational Intelligence, Proceedings of 11th International Work-Conference on Artificial Neural Networks (IWANN 2011)*, numéro 6692 de Lecture Notes in Computer Science, pages 276–283. Springer Berlin / Heidelberg, Malaga, Spain, 2011.

[9] G. CSARDI et T. NEPUSZ : The igraph software package for complex network research. *InterJournal*, Complex Systems, 2006.

[10] J.J. DAUDIN, F. PICARD et S. ROBIN : A mixture model for random graphs. *Statistics and Computing*, 18:173–183, 2008.

[11] G. DI BATTISTA, P. EADES, R. TAMASSIA et I.G. TOLLIS : *Graph Drawing : Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[12] P. EADES : A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[13] P. EADES et Q.W. FENG : Multilevel visualization of clustered graphs. *In* Stephen C. NORTH, éditeur : *Graph Drawing, Symposium on Graph Drawing, GD '96, Berkeley, California, USA, September 18-20, Proceedings*, volume 1190 de *Lecture Notes in Computer Science*, pages 101–112. Springer, 1996.

[14] P. EADES et M.L. HUANG : Navigating clustered graphs using force-directed methods. *Journal of Graph Algorithms and Applications*, 4(3):157–181, 2000.

[15] S. FORTUNATO : Community detection in graphs. *Physics Reports*, 486:75–174, 2010.

[16] S. FORTUNATO et M. BARTHÉLÉMY : Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007. doi :10.1073/pnas.0605965104 ; URL : `http://www.pnas.org/content/104/1/36.abstract`.

[17] T. FRUCHTERMAN et B. REINGOLD : Graph drawing by force-directed placement. *Software-Practice and Experience*, 21:1129–1164, 1991.

[18] Mark S. HANDCOCK, David R. HUNTER, Carter T. BUTTS, Steven M. GOODREAU et Martina MORRIS : *statnet : Software tools for the Statistical Modeling of Network Data*. Seattle, WA, 2003. Version 2.0.

[19] D. HAREL et Y. KOREN : Drawing graphs with non-uniform vertices. *In Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '02, pages 157–166, New York, NY, USA, 2002. ACM.

[20] I. HERMAN, G. MELANÇON et M. SCOTT MARSHALL : Graph visualization and navigation in information visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[21] P. D. HOFF, A. E. RAFTERY et M. S. HANDCOCK : Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97(460):1090–1098, December 2002.

[22] Pavel N. KRIVITSKY et Mark S. HANDCOCK : Fitting latent cluster models for networks with latentnet. *Journal of Statistical Software*, 24(5):1–23, 5 2008.

[23] R. MILO, N. KASHTAN, S. ITZKOVITZ, M. E. J. NEWMAN et U. ALON : On the uniform generation of random graphs with prescribed degree sequences. *eprint arXiv :cond-mat/0312028*, décembre 2003.

[24] M.E.J. NEWMAN : The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[25] M.E.J. NEWMAN : Finding community structure in networks using the eigenvectors of matrices. *Physical Review, E*, 74(036104), 2006.

[26] M.E.J. NEWMAN et M. GIRVAN : Finding and evaluating community structure in networks. *Physical Review, E*, 69:026113, 2004.

`Soumis au Journal de la Société Française de Statistique`
`File: rossi_villavialaneix_jsfds.tex, compiled with jsfds, version : 2009/12/09`
`date: 13 décembre 2011`

[27] A. NOACK : Energy models for graph clustering. *Journal of Graph Algorithms and Applications*, 11(2):453–480, 2007.

[28] A. NOACK : Modularity clustering is force-directed layout. *Physical Review E*, 79(026102), February 2009.

[29] A. NOACK et R. ROTTA : Multi-level algorithms for modularity clustering. *In SEA '09 : Proceedings of the 8th International Symposium on Experimental Algorithms*, pages 257–268, Berlin, Heidelberg, 2009. Springer-Verlag.

[30] P. PONS et M. LATAPY : Post-processing hierarchical community structures : Quality improvements and multi-scale view. *Theoretical Computer Science*, 412(8-10):892 – 900, 2011.

[31] R DEVELOPMENT CORE TEAM : *R : A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

[32] A. RAMACHANDRA RAO, R. JANA et S. BANDYOPADHYAY : A markov chain monte carlo method for generating random (0, 1)-matrices with given marginals. *Sankhya : The Indian Journal of Statistics, Series A*, 58(2):225–242, June 1996.

[33] J. REICHARDT et S. BORNHOLDT : Partitioning and modularity of graphs with arbitrary degree distribution. *Phys. Rev. E*, 76(1):015102, Jul 2007.

[34] J. M. ROBERTS JR. : Simple methods for simulating sociomatrices with given marginal totals. *Social Networks*, 22(3):273 – 283, 2000.

[35] F. ROSSI et N. VILLA-VIALANEIX : Optimizing an organized modularity measure for topographic graph clustering : a deterministic annealing approach. *Neurocomputing*, 73(7-9):1142–1163, 2010.

[36] S.E. SCHAEFFER : Graph clustering. *Computer Science Review*, 1(1):27–64, August 2007.

[37] M. SEIFI, J.L. GUILLAUME, M. LATAPY et B. LE GRAND : Visualisation interactive multi-échelle des grands graphes : application à un réseau de blogs. *In Atelier EGC 2010, Visualisation et Extraction de Connaissances*, Hammamet, Tunisie, 2010.

[38] Q.D. TRUONG, T. DKAKI et P.J. CHARREL : An energy model for the drawing of clustered graphs. *In Proceedings of Vème colloque international VSST*, Marrakech, Maroc, 21-25 octobre 2007.

[39] D. TUNKELANG : *A Numerical Optimization Approach to General Graph Drawing*. Thèse de doctorat, School of Computer Science, Carnegie Mellon University, January 1999. CMU-CS-98-189.

[40] U. von LUXBURG : A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[41] X. WANG et I. MIYAMOTO : Generating customized layouts. *In* Franz BRANDENBURG, éditeur : *Graph Drawing*, volume 1027 de *Lecture Notes in Computer Science*, pages 504–515. Springer Berlin / Heidelberg, 1996.

[42] H. ZANGHI, C. AMBROISE et Miele V. : Fast online graph clustering via erdös-rényi mixture. *Pattern Recognition*, 41:3592–3599, 2008.

Soumis au Journal de la Société Française de Statistique
File: rossi_villavialaneix_jsfds.tex, compiled with jsfds, version : 2009/12/09
date: 13 décembre 2011